Applications Services

Design Review

Thor DP3

17 November 97
Version 2.1

1. Application Services

1.1 Application Services Introduction

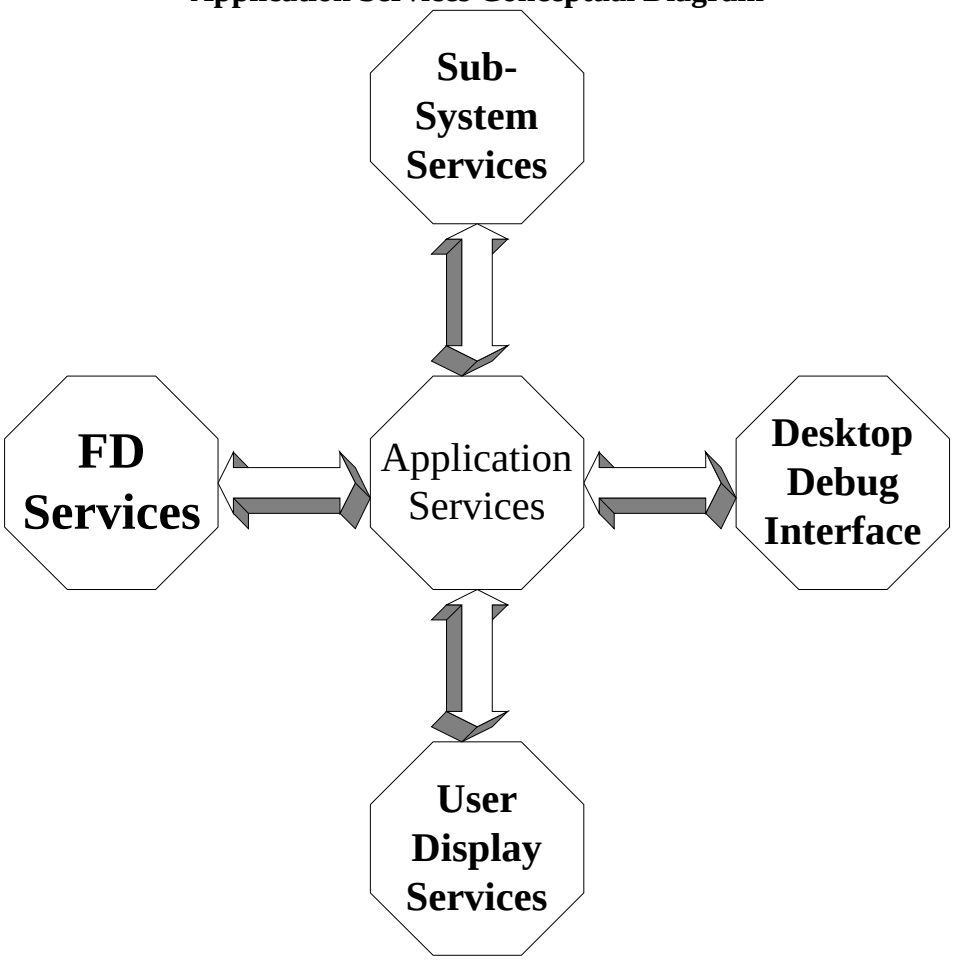
1.1.1 Application Services Overview

Application Services (CM symbol ASV) is a collection of Object Oriented classes that encapsulate APIs providing an interface between User or Systems Applications and Systems Services. This protects applications from any changes in the underlying Systems Services interface, as well as protecting system services from changes in CLCS application tools. Subsequent adaptations to System Services can thus be accommodated through Application Services without altering the applications themselves.

Application Services CSCs are:

- FD Services (Thor)
- Constraint Management Services (Thor)
- User Display Services (Thor)
- Math Model Services (post-Redstone)
- End Item Manager Services (Thor)
- Prerequisite Control Services (Thor)
- Test Application Script (TAS) Services (post-Thor)
- User Advisory Services (post-Thor)
- Sub-System Services (Thor)

Application Services Conceptual Diagram



1.1.2 Application Services Operational Description

Application Services provides applications with an object-oriented interface to underlying services. Application Services implements a C++ layer that provides data type checking similar to the GOAL language.

1.2 Application Services Specifications

1.2.1 Application Services Ground-rules

1. All Application Services will provide C++ APIs, using the approach agreed to with User Applications on 6/4/97.
2. Application Services will be implemented in C++ using single inheritance and minimal reference (pointer) usage to support possible future transitions to languages such as Java.

1.2.2 Applications Services Common Functional Requirements

This section defines requirements common to all Application Services APIs.

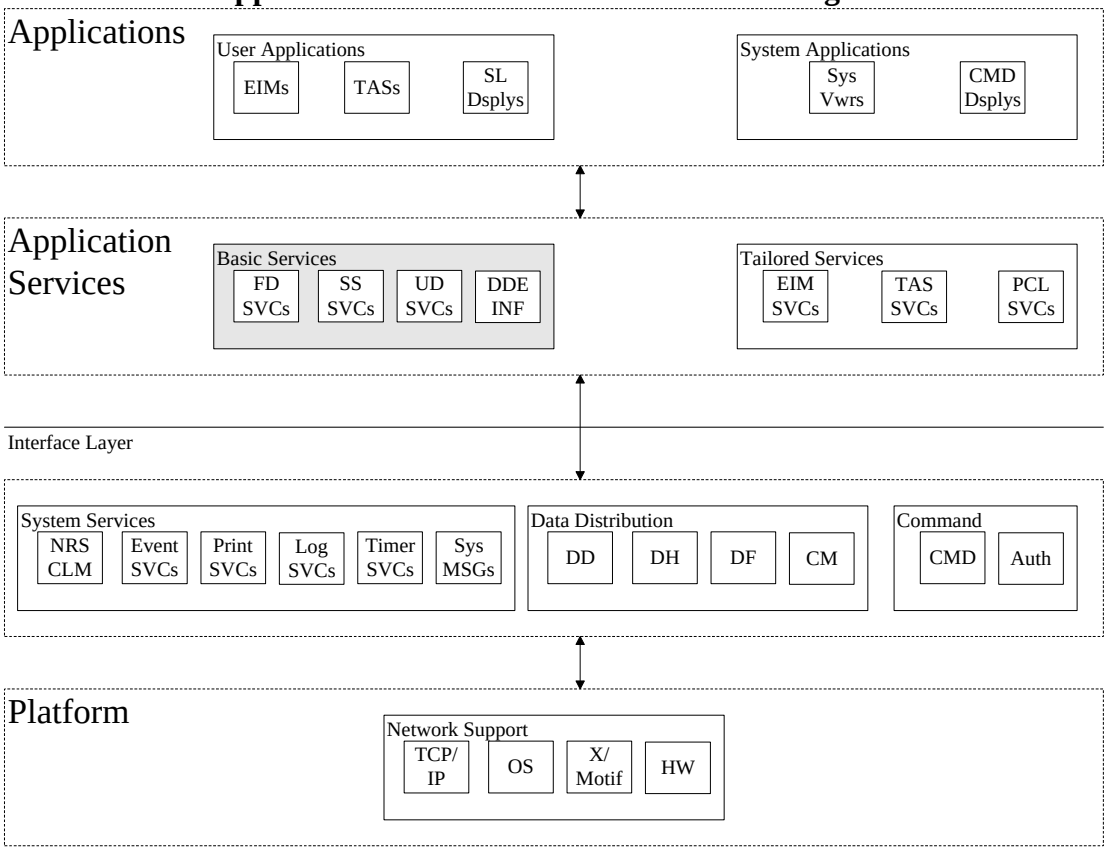
1. Application Services shall return status to the calling application on the success or failure of every API call made to an Application Service.
2. When an Application Services API call fails, the Application Service shall send an error message to the System Message Writer specifying the reason for any failure condition.

1.2.3 Application Services Performance Requirements

Thor performance requirements are listed separately for each CSC.

1.2.4 Application Services Interfaces Data Flow Diagrams

Application Services External Data Flow Diagram



2. Function Designator (FD) Services

2.1 FD Services Introduction

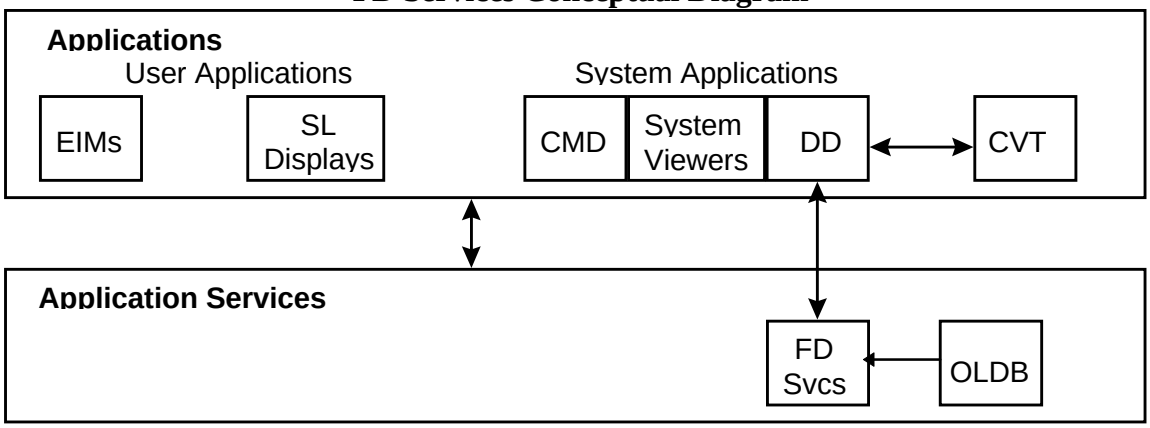
2.1.1 FD Services Overview

"This service provides Function Designator (FD) measurement and stimulus data." FD Services shall perform the following services:

- Provide APIs allowing applications to access [1]processed value FDs.
- Provide APIs allowing applications to access queued multi-sample FDs.
- Provide APIs allowing applications to write FDs.

Function Designator (FD) Services provides applications a set of type-safe common access methods for reading and writing FD data values. These data values are stored in the OLDB, the Current Value Table (CVT), and the algorithm tables (per C. King, 5/17/97).

FD Services Conceptual Diagram



2.1.2 FD Services Operational Description

FD Services provides a thin layer between applications and systems software. This thin layer ensures that changes made to underlying systems layers do not cause impacts across the range of user applications, and that changes in user applications tools do not cause impacts to systems services. FD Services provides a C++ implementation of the types and type checking available in GOAL.

2.2 FD Services Specifications

2.2.1 FD Services Ground-rules

- FD Services performs all FD reads (including pseudo FDs) via Data Distribution.
- FD Services performs all FD writes via Command Support.
- The OLDB will be read-only.
- There will only be one OLDB per target CLCS "set".
- FD Services will use the OLDB provided by System Build CSCI.
- FD Services makes no distinction between pseudo-FDs and real FDs.
- FD Services will provide for all FD accesses an atomic set of value, time and health.

[1] Counts are not contained in the Current Value Table. Having access to counts may be an issue for a later delivery.

- For each Time Homogeneous Data Set (THDS) present in the TCID the OLDB will contain a THDS FD.
- THDS FDs will be designated by a specific and unique FD type designation in the OLDB.
- Enumerated FDs will be designated by a specific and unique FD type designation in the OLDB.
- FD Services will inherit objects from Application Services Subsystem Services for interactions with the Command Interface.

2.2.2 FD Services Functional Requirements

The following paragraphs define the FD Services functional requirements:

- Read FDs
- Read Queued Multi-Sample FDs
- Write [2] FDs

Read FDs

1. FD Services shall provide an API to read the current data value for any valid FD.
2. FD Services shall provide an API to read the current value of an analog FD in engineering units for the analog FD types defined in CLCS System Level Specification, 84K00200-000, pre-release 1, dated 15 April 1997
3. FD Services shall provide an API to read the current value of a digital pattern FD.
4. FD Services shall provide an API to read the current value of a discrete FD for the following engineering unit types:
   - 4.1. Open / Close
   - 4.2. True / False
   - 4.3. Wet / Dry
   - 4.4. On / Off.
5. FD Services shall provide an API to read the current value of a discrete FD in raw format.
6. FD Services shall provide an API to read the time of the last change in value of the FD.
7. FD Services shall provide an API to read the health status of the last change of an FD.
8. FD Services shall provide the capability to access all current data attributes from the OLDB for any valid FD.[3]
9. FD Services shall provide an API to read an FDs current value, time of last value change, and health status in a single request.
10. FD Services shall provide an API to read calibration data as FDs from the Gateways.
11. FD Services shall provide the capability to sequentially read all FD information from the OLDB.
12. FD Services shall provide an API to read the ASCII text string associated with a specified health reason or warning code.
13. FD Services shall provide the capability to read the current value of a enumerated FD.
14. FD Services shall provide the capability to read Time Homogeneous Data Sets (THDS) of FDs.
15. FD Services shall provide an API to access all current data attributes from the CVT for any valid FD.

Read Queued Multi-Sample FDs

1. FD Services shall provide Queued multi-sample service to applications for any valid FD.
2. FD Services shall provide an API to identify that an FD should be delivered via queued service (multi-sample registration and de-registration).
3. FD Services shall provide an API to access every change value in time sequential fashion.
4. FD Services shall provide an API to read the next value of a multi-sample FD.
5. FD Services shall provide an API to read the next N values of a multi-sample FD.
6. FD Services shall provide an API to clear all queued samples pending for the application.
7. FD Services shall provide an API to notify user applications when multi-sample queued data is available for a relevant FD.

---

[2] There is no distinction between real and pseudo FDs in the non-command CLCS design. Command Management understands that pseudo-FDs are not associated with a gateway and makes a Data Distribution API call to distribute the change value.

[3] Additions to the OLDB for Thor include fields to support PCL, and Gateway change data output types.

8. FD Services shall provide an API to start queued multi-sample delivery by FD.
9. FD Services shall provide an API to stop queued multi-sample delivery by FD.

Write FD

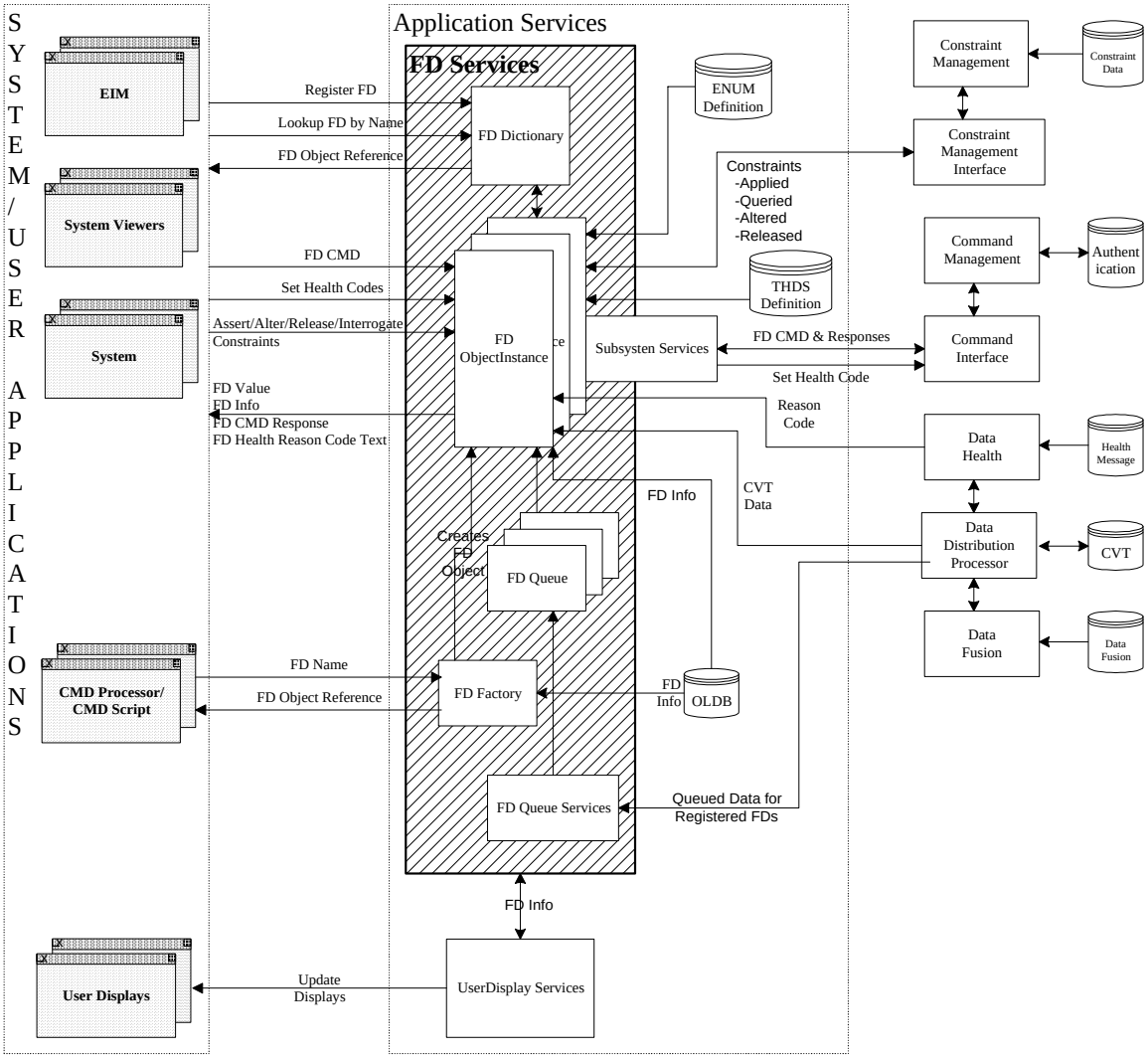
1. FD Services shall provide an API to write values to analog output FD's.
2. FD Services shall provide an API to write a value to discrete output FD's.
3. FD Services shall provide an API to write discrete output FD's using the following discrete data types: OPEN, CLOSE, TRUE, FALSE, WET, DRY, ON, OFF.
4. FD Services shall provide an API to write a value to digital pattern output FD's.
5. FD Services shall provide an API to write a time value in a TBD time format.
6. FD Services shall provide an API to write failure and warning reason codes for an FD.
7. FD Services shall provide an API to write failure and warning reason codes for a list of FDs.
8. FD Services shall provide an API to write failure and warning reason codes for a predefined list of FDs specified by a "group name".

2.2.3 FD Services Performance Requirements

- No specific performance requirements have been established for the FD Services Thor delivery.

2.2.4 FD Services Interface Data Flow Diagrams

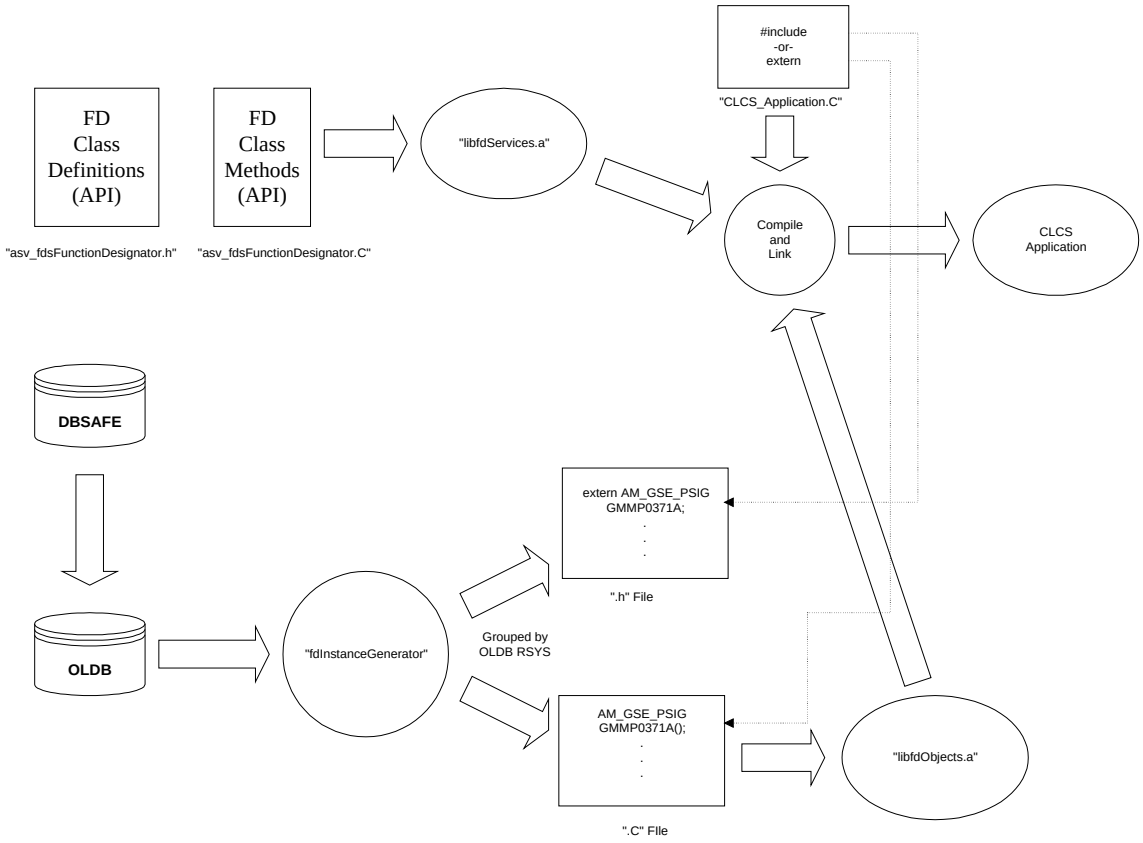
FD Services External Data Flow Diagram



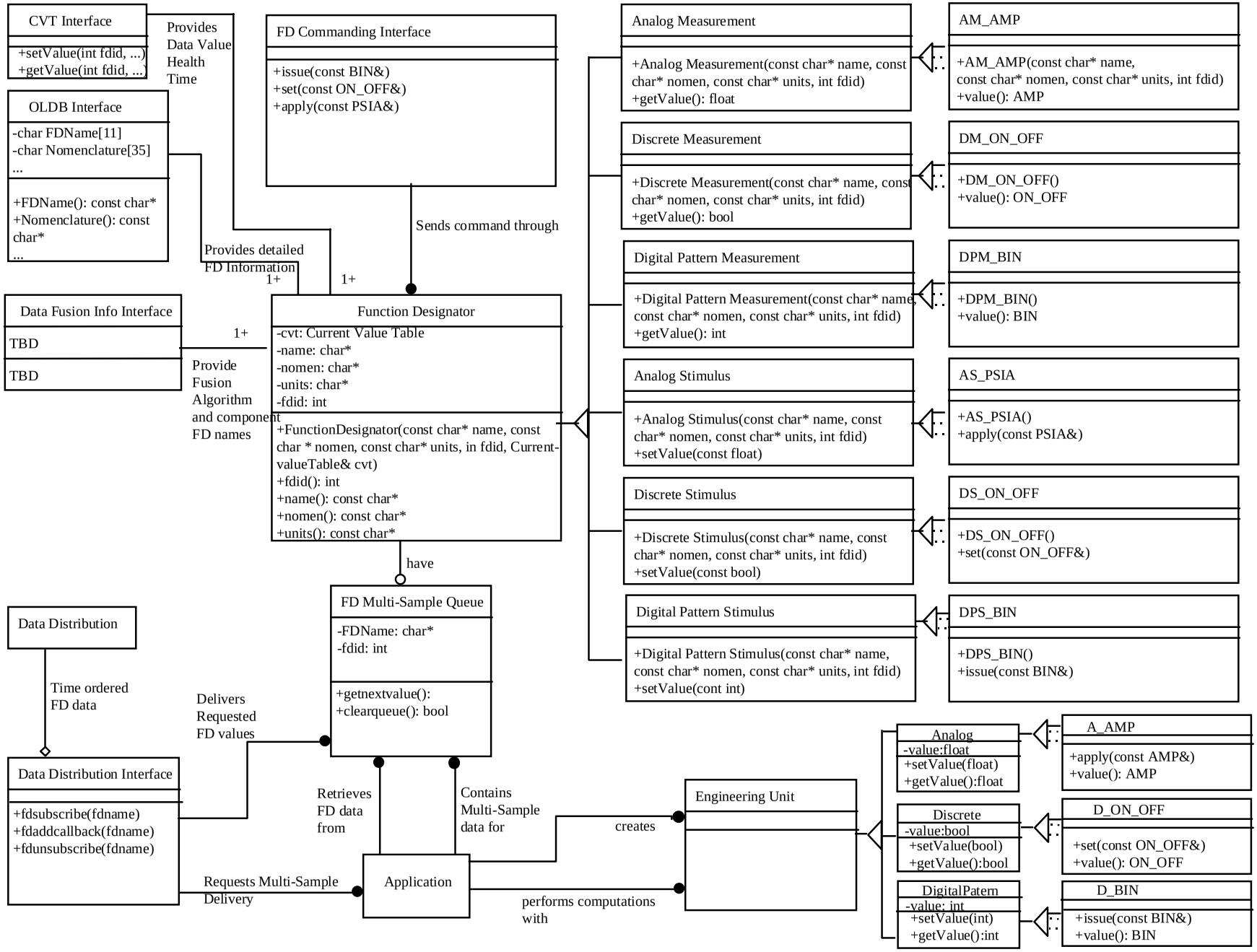
2.3 FD Services Design Specification

The FD Services CSC is object oriented. FD Services will be implemented in C++ as a collection of classes contained in a library that is linked into software applications wishing to utilize the FD Services interface. FD Services supports creating a library that contains FD object instances for a particular TCID. FD services provides an executable that is maintained in the System build process that is utilized as part of the TCID build process to generate the FD object instance library.

FD Object Instantiation and Compialiation Process:



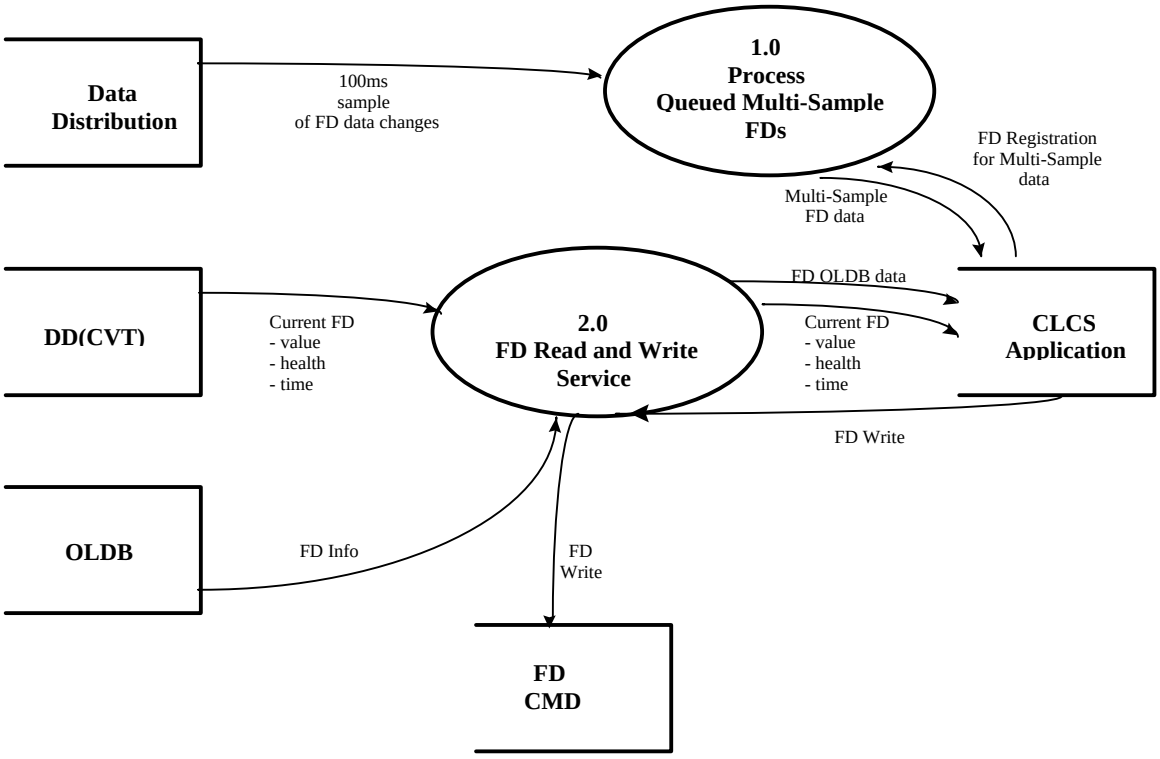
The following page provides an object class diagram for FD Services.


CVT Interface
+setValue(int fdid, ...)
+getValue(int fdid, ...
Provides
Data Value
Health
Time
FD Commanding Interface
+issue(const BIN&)
+set(const ON_OFF&)
+apply(const PSIA&)
OLDB Interface
-char FDName[11]
-char Nomenclature[35]
...
+FDName(): const char*
+Nomenclature(): const char*
...
Sends command through
Provides detailed
FD Information
1+
1+
Data Fusion Info Interface
TBD
TBD
1+
Provide
Fusion
Algorithm
and component
FD names
Function Designator
-cvt: Current Value Table
-name: char*
-nomen: char*
-units: char*
-fdid: int
+FunctionDesignator(const char* name, const char * nomen, const char* units, in fdid, Current-valueTable& cvt)
+fdid(): int
+name(): const char*
+nomen(): const char*
+units(): const char*
have
Analog Measurement
+Analog Measurement(const char* name, const char* nomen, const char* units, int fdid)
+getValue(): float
AM_AMP
+AM_AMP(const char* name, const char* nomen, const char* units, int fdid)
+value(): AMP
Discrete Measurement
+Discrete Measurement(const char* name, const char* nomen, const char* units, int fdid)
+getValue(): bool
DM_ON_OFF
+DM_ON_OFF()
+value(): ON_OFF
Digital Pattern Measurement
+Digital Pattern Measurement(const char* name, const char* nomen, const char* units, int fdid)
+getValue(): int
DPM_BIN
+DPM_BIN()
+value(): BIN
Analog Stimulus
+Analog Stimulus(const char* name, const char* nomen, const char* units, int fdid)
+setValue(const float)
AS_PSIA
+AS_PSIA()
+apply(const PSIA&)
Discrete Stimulus
+Discrete Stimulus(const char* name, const char* nomen, const char* units, int fdid)
+setValue(const bool)
DS_ON_OFF
+DS_ON_OFF()
+set(const ON_OFF&)
Digital Pattern Stimulus
+Digital Pattern Stimulus(const char* name, const char* nomen, const char* units, int fdid)
+setValue(cont int)
DPS_BIN
+DPS_BIN()
+issue(const BIN&)
Data Distribution
Time ordered
FD data
FD Multi-Sample Queue
-FDName: char*
-fdid: int
+getnextvalue():
+clearqueue(): bool
Delivers
Requested
FD values
Data Distribution Interface
+fdsubscribe(fdname)
+fdaddcallback(fdname)
+fdunsubscribe(fdname)
Retrieves
FD data
from
Contains
Multi-Sample
data for
Requests Multi-Sample
Delivery
Application
creates
performs computations
with
Engineering Unit
Analog
-value:float
+setValue(float)
+getValue():float
A_AMP
+apply(const AMP&)
+value(): AMP
Discrete
-value:bool
+setValue(bool)
+getValue():bool
D_ON_OFF
+set(const ON_OFF&)
+value(): ON_OFF
DigitalPatern
-value: int
+setValue(int)
+getValue():int
D_BIN
+issue(const BIN&)
+value(): BIN

2.3.1 FD Services Detailed Data Flow

This diagram provides a pictorial representation of the data flow between applications, Data Distribution, Data Health, Data Fusion, and FD Services objects.

Detailed Data Flow for FD Services



2.3.2 FD Services External Interfaces

2.3.2.1 FD Services Message Formats

The following are the System Messages output by the FD Services CSC.

Message Number = ASV_FDS_CVT_ATTACH_ERROR
Message Group = ASV
Severity = Error (possible levels: Warning, Error, Informational)

ASV FD Services unable to access CVT for Process Name #ARGUMENT1# reason #ARGUMENT2# - #ARGUMENT3#

ARGUMENT1 = ASCII character string representing UNIX process name.
ARGUMENT2 = unsigned integer representing UNIX error number value.
ARGUMENT3 = ASCII character string that describes error condition corresponding to ARGUMENT2 provided by UNIX.

Help Information Content:

During application initialization the FD Services interface could not properly attach to Data Distribution's CVT shared memory area. Check to make sure that Data Distribution process is running. Also, verify that the OLDB flat file is present (because Data Distribution needs this file to generate the CVT).

Detailed Information:

ARGUMENT3 will provide a description of the reason for the error.

Message Number = ASV_FDS_QUEUED_FD_INTERFACE_INIT_ERROR
Message Group = ASV
Severity = Error (possible levels: Warning, Error, Informational)

ASV FD Services Queued Multi-Sample FD interface initialization error for process name #ARGUMENT1# reason #ARGUMENT2# - #ARGUMENT3#

ARGUMENT1 = ACSII character string representing UNIX process name.
ARGUMENT2 = unsigned integer representing UNIX error number value.
ARGUMENT3 = ASCII character string that describes error condition corresponding to ARGUMENT2 provided by UNIX.

Help Information Content:

A request from an application to initialize a Queued Multi-Sample FD interface failed. One possible cause of this error is that Data Distribution did not provide access to the queuing service; please see the system message file for other errors that occurred during this period (particularly from Data Distribution).

Detailed Information:

ARGUMENT3 will provide a description of the reason for the error. The termination of the Data Distribution CVT server System Application would cause the interface closure error.

Message Number = specified system message number constant
Message Group = ASV
Severity = Error (possible levels: Warning, Error, Informational)

ASV FD Services Queued Multi-Sample FD interface error for process name #ARGUMENT1# reason #ARGUMENT2# - #ARGUMENT3#

ARGUMENT1 = ASCII character string representing UNIX process name.
ARGUMENT2 = unsigned integer representing UNIX error number value.
ARGUMENT3 = ASCII character string that describes error condition corresponding to ARGUMENT2 provided by UNIX.

Help Information Content:

During Queued Multi-Sample FD operations a UNIX system error was detected. The error could be either a failed read interface error or a unexpected closure of interface error. This usually would occur if Data Distribution terminated and restarted (a socket was changed). Please see the system message log around that time period for any Data Distribution activity.

Detailed Information:

ARGUMENT3 will provide a description of the reason for the error. The termination of the Data Distribution CVT server System Application would cause the interface closure error.

Message Number = specified system message number constant
Message Group = ASV
Severity = Error (possible levels: Warning, Error, Informational)

ASV FD Services encountered an error reading from the OLDB file, #ARGUMENT1# reason #ARGUMENT2# - #ARGUMENT3#

ARGUMENT1 = ASCII character string representing UNIX process name.
ARGUMENT2 = unsigned integer representing UNIX error number value.
ARGUMENT3 = ASCII character string that describes error condition corresponding to ARGUMENT2 provided by UNIX.

Help Information Content:

FD Services was unable to access data in the OLDB table. The table does not exist in the path specified or does not have adequate permissions. Verify that the OLDB flat file exists in the proper directory, that read permission has been granted for that file, and that the TCID is correct.

Detailed Information:

ARGUMENT3 will provide a description of the reason for the error.

2.3.2.2 FD Services Display Formats

FD Services does not provide any displays.

2.3.2.3 FD Services Input Formats

There are no language-like interfaces provided by FD Services.

2.3.2.4 Recorded Data

FD Services does not record data nor initiate data recording.

2.3.2.5 FD Services Printer Formats

FD Services does not provide printed information.

2.3.2.6 Inter-process Communications

Inter-process communications takes place between FD Services and Data Distribution. The Data Distribution CSC communicates FD data to the FD Services CSC through a UNIX shared memory area called the Current Value Table (CVT).

The CVT consists of an index area that maps both FD names and FD-IDs to the data portion of the CVT containing current FD value, the FD's health bits, the time of the last data change, and the time of the last health bit change.

The CVT communication is further described in the CLCS Data Distribution to FD Services Interface Definition Document.

2.3.2.7 FD Services External Interface Calls

The CLCS FD Services Interface Definition Document describes the data sent between the FD Services CSC and CLCS applications via a calling mechanism.

2.3.2.8 FD Services Table Formats

FD Services utilizes three table internally that are provided by an outside source; these tables are the OLDB flat file, the THDS definition flat file, and the Enumeration Definition flat file provided by System Build CSCI. Table format is TBD. An ASCII table will be provided by System Build for Redstone.

2.3.3 FD Services Test Plan

FD Services system-level tests may be run in either or both the IDE or SDE environments. These tests are run on the basic HCI, CCP or DDP platforms. There are no specific hardware configurations required. The minimal applications software configuration includes the Data Distribution server and whatever programs and files are necessary to have the Data Distribution server running. FD Services testing also requires a CLCS application or a CLCS like-application test tool that exercises the FD read, FD write, OLDB read, and Queued Multi-Sample FD services.

2.3.3.1 FD Services / Read FDs

Test Objective:
Test and / or Regression test the following FD Svcs capabilities to read FD value and parameter data. This also verifies this interface returns status to the calling application.

1. Read value, time, & health of FDs (DDP, CCP, & CCW/S) for both raw and EU types:(2.2.2.1.1, 2.2.2.1.7, 2.2.2.1.8,)
    - 1.1. Analog (one or more of each analog data type - TBD data types) (2.2.2.1.2)
    - 1.2. Discrete (one or more of each discrete data type - 8 data types) (2.2.2.1.4, 2.2.2.1.6)
    - 1.3. Digital Pattern (one or more of each of the following data types: Hex, Octal, BCD, Enumerated Type) (2.2.2.1.3, 2.2.2.1.14)
    - 1.4. Calibration data as FDs from the Gateways (2.2.2.1.11)
2. Verify API can read all current data attributes from the CVT for any valid FD. (2.2.2.1.10)
3. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

Using known data from a Sim G/W for STS-TBD, use a TBD test application to make calls to FD Svcs on each subsystem type (DDP, CCP, CCW/S). Use a CVT viewer (CVT_Look) or System Viewer (FD Viewer) to verify the data matches for that time sample.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging / Network Services
- Data Source (PC-GOAL data for STS-TBD feeding into SIM G/W) or equivalent (to provide FD data values in CVT)
- Search function for Data Source (fast-forward, reverse, stop, pause) for finding FD data changes
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS

- TBD CVT Viewing tool for DDP, CCP, and CCWS CVTs, viewable on a CCWS (CVT_LOOK from DD or other tool)
- Tool or test application to call FD Svcs (DNAV or equivalent with pre-coded scripts)

Requirement(s):

- SLS Requirement(s): 2.2.3.1.3, 2.2.3.1.4, 2.2.5.2.1, 2.2.5.2.2, 2.2.5.2.3.
- FD Services Requirement(s): 2.2.2.1.1, 2.2.2.1.2, 2.2.2.1.3, 2.2.2.1.4, 2.2.2.1.6, 2.2.2.1.7, 2.2.2.1.8, 2.2.2.1.9, 2.2.2.1.10, 2.2.2.1.11, 2.2.2.1.16, 2.2.2.1.14.
- RTC Applications (User) Requirement(s): 4.2.2.1-7

2.3.3.2 FD Services / Read Queued FDs

Test Objective:

Test and / or Regression test the following FD Svcs capabilities to read FD value and parameter data. This also verifies this interface returns status to the calling application.

1. Read value, time, & health of Queued FDs for both raw and EU types: (2.2.2.2.1, 2.2.2.2.2, 2.2.2.2.3)
    - 1.1. Analog (one or more of each analog data type - TBD data types)
    - 1.2. Discrete (one or more of each discrete data type - 8 data types)
    - 1.3. Digital Pattern (one or more of each of the following data types: Hex, Octal, BCD, Enumerated Type)
2. Read next value of a multi-sample FD. (2.2.2.2.4)
3. Read next N values of a multi-sample FD. (2.2.2.2.5)
4. Clear all queued data pending for an application (2.2.2.2.6)
5. Notify user applications when multi-sample queued data is available (2.2.2.2.7)
6. Start queued service to an FD. (2.2.2.2.8)
7. Stop queued service to an FD. (2.2.2.2.9)
8. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

Using known data from a Sim G/W for STS-TBD, use a TBD test application to make calls to FD Svcs on each subsystem type (DDP, CCP, CCW/S). Use a CVT viewer (CVT_Look) or System Viewer (FD Viewer) to verify the data matches for that time sample.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging / Network Services
- Data Source (PC-GOAL data for STS-TBD feeding into SIM G/W) or equivalent
- Search function for Data Source (fast-forward, reverse, stop, pause) for finding FD data changes
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS
- TBD CVT Viewing tool for DDP, CCP, and CCWS CVTs, viewable on a CCWS (CVT_LOOK from DD or other tool)
- Tool to examine CVT values
- Tool or test application to call FD Svcs (DNAV or equivalent with pre-coded scripts)

Requirement(s):

- SLS Requirement(s): 2.2.3.1.3, 2.2.3.1.4, 2.2.5.2.1, 2.2.5.2.2, 2.2.5.2.3.
- FD Services Requirement(s): 2.2.2.1.1, 2.2.2.1.2, 2.2.2.1.3, 2.2.2.1.4, 2.2.2.1.6, 2.2.2.1.7, 2.2.2.1.8, 2.2.2.1.9, 2.2.2.1.10, 2.2.2.1.11, 2.2.2.1.16, 2.2.2.1.14.
- RTC Applications (User) Requirement(s): 4.2.3.1-5

2.3.3.3 FD Services / Write FDs

Test Objective:

Test and / or Regression test the following FD Svcs capabilities to write FD value and parameter data:

1. Write value of FDs at a CCW/S:
    - 1.1. Analog (one or more of each analog data type - TBD data types) (2.2.2.3.1)

1.2. Discrete (one or more of each discrete data type - 8 data types) (2.2.2.3.2, 2.2.2.3.3)
1.3. Digital Pattern (one or more of each of the following data types: Hex, Octal, BCD, Enumerated Type) (2.2.2.3.4)
1.4. Time values in the following formats (JTOY, CDT/MET, others?) (2.2.2.3.5)
2. Write failure and warning reason codes for an FD. (2.2.2.3.6)
3. Write failure and warning reason codes for a list of FDs. (2.2.2.3.7)
4. Write failure and warning reason codes for a list of FDs by "group name". (2.2.2.3.8)
5. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:
The test conductor will use a Command-provided input screen to write the FD value for each FD type (>50 analog types, 8 discrete types, Hex, Octal, BCD, and enum digital types). On another window on the same W/S, the test conductor will use CVT_Look or an equivalent tool to examine the change in the FD's value in the CVT. The test conductor will use the same configuration to write the Health status for an FD (warning, then failure). A pre-defined list of FDs (in TBD file) will have its health status changed to warning (and back) and to failure (and back). Each FD in the list will be examined. A pre-defined, named list will have its health status changed to warning (and back) and to failure (and back). Each FD in the list will be examined.

Required Test Configuration / Dependencies:
- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN and DCN
- Reliable Messaging / Network Services
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS
- **Command Management** or equivalent mechanism to send Writes to Gateways
- Command Processor or equivalent to input FD writes at CCWS
- TBD CVT Viewing tool for CCWS to examine CVT values (CVT_LOOK from DD or other tool)

Requirement(s):
- SLS Requirement(s): 2.2.3.3
- FD Services Requirement(s): 2.2.2.3.1, 2.2.2.3.2, 2.2.2.3.3, 2.2.2.3.4, 2.2.2.3.5, 2.2.2.3.6, 2.2.2.3.7, 2.2.2.3.8.

  RTC Applications (User) Requirement(s): 4.2.1.1, 4.2.1.2, 4.2.1.3, 4.2.3.6.

2.3.3.4 FD Services OLDB reads

Test Objective:
Test and / or Regression test the following FD Svcs capabilities:
1. Sequentially read all FD information from the OLDB. (2.2.2.1.12)
2. Read all current data attributes from the CVT for any valid FD. (2.2.2.1.16)
3. Query the OLDB by FDID or FD name. (user rqmt 4.2.5.1)
4. Query any piece of information stored in the OLDB for a particular FD. (user rqmt 4.2.5.2)
5. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:
From a TBD Viewer on a CCWS, the test conductor will select an FD by FDID and request all OLDB data for that FD. Then the test conductor will select an FD by FD name and request all OLDB data for that FD. The data on the display will be compared to that printed from the OLDB.

Required Test Configuration / Dependencies:
- Ops CM to Configure Flow Zone
- CCWS platform
- Thor TCID, with Fusion FDs and pseudo FDs defined
- TBD CVT Viewing tool for DDP, CCP, and CCWS
- Tool to examine OLDB values (TBD)

Requirement(s):
- SLS Requirement(s): N/A
- FD Services Requirement(s): 2.2.2.1.12, 2.2.2.1.16.

- RTC Applications (User) Requirement(s): 4.2.5.1, 4.2.5.2

2.3.3.5 FD Health Services (User requirements)

Test Objective:
Test the following FD Health Svcs capabilities to determine the health of an FD and to read the detailed health status for FDs:

1. Read FD's health status (OK, FAILED, WARNING): (4.2.4.4.1)
    - 1.1. For one (or more) FDs for each Gateway type supported in Thor (Consolidated SDS, others?)
    - 1.2. For Analog, Discrete, and Digital Pattern FD types (one or more of each data type)
    - 1.3. For Pseudo FDs (?)
    - 1.4. For Time value FDs (?)
2. Read the detailed health status for one or more FDs: (4.2.4.4.2)
    - 2.1. Was the last value change or refresh data?
    - 2.2. Is processing active or inhibited for this FD?
    - 2.3. Is gateway group processing active or inhibited for this FD?
    - 2.4. Is Engineering active or inhibited for this FD? (same as #7? - Check with Rich Ikerd.)
    - 2.5. Is the data path associated with this FD active or inhibited?
    - 2.6. Is application advisory notification active or inhibited for this FD?
    - 2.7. Is engineering bypass active or inhibited for this FD?
3. Verify each API call returns correct status (success only).

Test Approach Summary:
The test conductor will read the health status for an FD (nominal = OK) using a Health Viewer (or equivalent) at a CCWS. From another window on the CCWS, the test conductor will change the health status to Warning, then to Failed, then back to OK.
The test conductor will read the detailed health status using the Data Health Viewer at the CCWS. Note: There may be limited detailed health status available at CIT time. This may require a delta CIT prior to System Test, or may require additional testing during System Test.
Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100baseT) and DCN (FDDI)
- Reliable Messaging or equivalent / Network Services
- Thor TCID, with Fusion FDs and pseudo FDs defined
- **Gateways** (Consolidated SDS, GSE, LDB, others?)
- Data Distribution / CVT on DDP, CCP, and CCWS
- **Data Distribution** must provide a new bit specifying whether each FD data value is change data or refresh data (4.2.4.4.2.1).
- **Data Health** with full detailed health status for FDs.
- **TBD Health Viewer** or equivalent mechanism to set FD health status / detailed status at CCWS

Requirement(s):

- SLS Requirement(s): 2.2.5.2.1-4, 2.2.5.2.1.8
- FD Services Requirement(s): N/A
- RTC Applications (User) Requirement(s): 4.2.4.4.1, 4.2.4.4.2.1-7, 4.2.4.4.4

2.3.3.6 FD Processing Attributes (User requirements 4.2.1.1, 4.2.1.4, 4.2.1.7-10)

Test Objective:
Test the following FD processing capabilities:

1. Change the sample rate of a GSE FD. (4.2.4.3.1)
2. Change the responsible subsystem for an FD. (4.2.4.3.5)
3. Activate, then inhibit measurement processing for an FD. (4.2.4.3.6)
4. Read the current sample rate of an FD. (4.2.4.3.8)
5. Read the hardware address of an FD. (4.2.4.3.11)
6. Verify each API call returns correct status (success only).

Test Approach Summary:

The test conductor will use a TBD Viewer that can display the following Gateway-related data for an FD:

1. sample rate
2. responsible subsystem (RSYS)
3. measurement processing status (active / inhibited)
4. hardware address

The test conductor will use a Command-provided input screen to change each Gateway processing value for the FDs tested. The results in the TBD Viewer will be compared to the planned results.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS
- **GSE Gateway, Consolidated SDS Gateway**
- **Command Management** or equivalent mechanism to send Writes to Gateways
- **Command Processor** or equivalent to input Gateway commands at CCWS
- **TBD Viewing tool** for CCWS to examine Gateway-related FD processing data

Requirement(s):

- SLS Requirement(s):
- FD Services Requirement(s): N/A
- RTC Applications (User) Requirement(s): 4.2.4.3.1, 4.2.4.3.5, 4.2.4.3.6, 4.2.4.3.8, 4.2.4.3.11

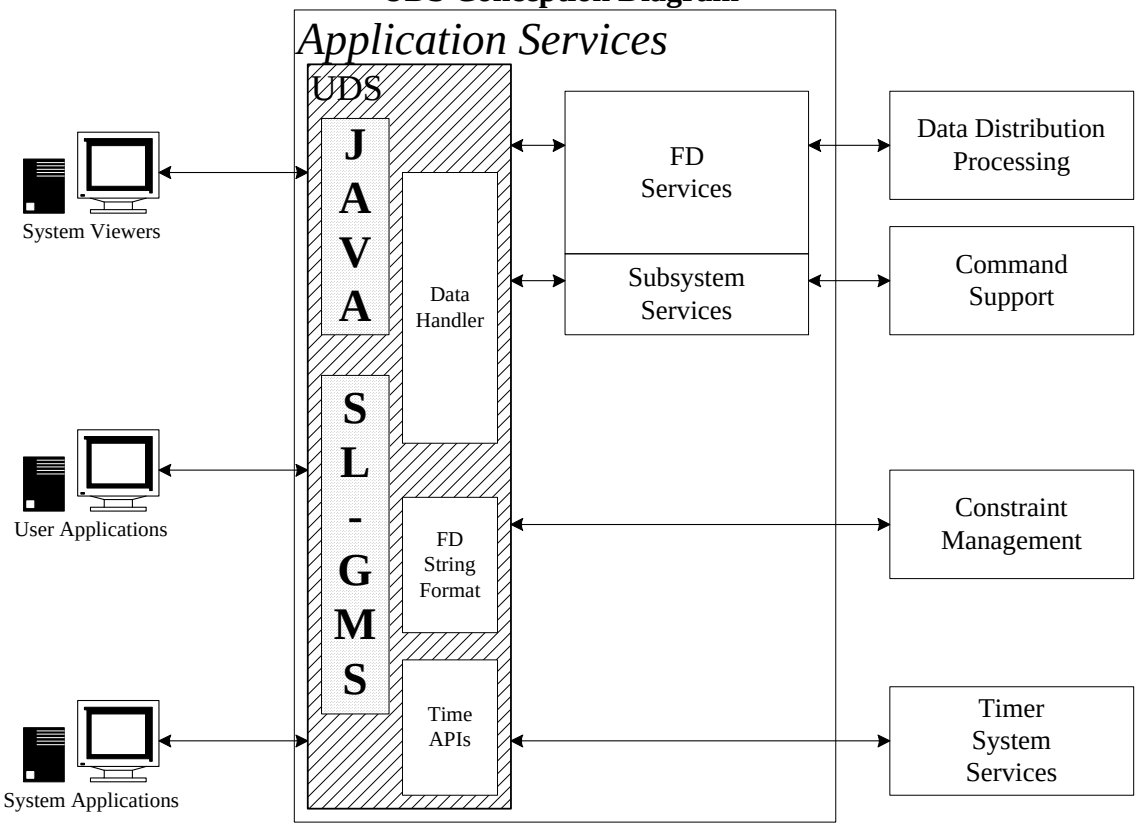
3. User Display Services (UDS)

3.1 User Display Services Introduction

3.1.1 User Display Services Overview

User Display Service (UDS) provides the capability for applications to communicate with the displays. UDS is a series of common routines provided to application programs giving them the ability to update visual displays with current Function Designator (FD) values, health, and time. UDS will also provide for System Viewers and user displays the capability to get current FD information.

UDS Conception Diagram



3.1.2 User Display Services Operational Description

User application displays will register with UDS all interested FD names. UDS will provide the displays with the latest changed values, health, and time of the FD from the Current Value Table (CVT) via FD Services. To stop updating the FD, user application displays will notify UDS to stop retrieving the latest data.

UDS will give applications the capability to call System Viewers to retrieve all of the latest FD information provided by FD Services.

UDS will interface with System Viewers, System Applications, and User Applications. UDS is the layer of Application Services to separate the user interface from the System Services. UDS provides a collection of FD as a data handler to manage a list of FDs. The data handler will provide the ability to get the latest values from the CVT through FD Services to perform updates to the User Display. Multiple types of Data Handlers will be used to provide information for each system viewer, as well as the ability to get different types of data to the display.

UDS will also provide a layer common routines for Java displays to call the underlying C++ APIs. UDS will also provide notification to the viewers of a transition of constraints on an FD. Time display functions for SL-GMS displays will also be supported, as well as the ability to start an EIM from a display.

3.2 User Display Services Specifications

3.2.1 User Display Services Ground-rules

- UDS will interface to SL-GMS v.5.3 as the Thor CLCS display development tool.
- UDS will interface to Java Development Kit (JDK) v1.1.2.
- User application and System Viewers displays will use UDS to access current data from the CVT and OLDB.
- UDS will **NOT** provide queued (multi-sample) nor historical FD data.
- UDS will be object oriented, written in C++ and JDK v1.1.2.
- UDS performance is based upon data update rates to the local CVT.

3.2.2 User Display Services Functional Requirements

1. UDS shall provide a mechanism allowing an FD-related widget on an SL-GMS display to access the current FD data from the CVT / OLDB without callback programming.
2. UDS shall provide an API allowing System Viewers or user application displays access FD related information from the CVT/OLDB.
3. UDS shall provide a way for applications to access common library dialog functions to:
    - 3.1. Provide a two step modeless pop up window.
        - 3.1.1. Provide a **CANCEL** button.
        - 3.1.2. Provide an **EXECUTE** button.
    - 3.2. Provide a text display modeless pop up window for command formatting errors.
        - 3.2.1. Provide an **OK** button.
4. UDS shall provide an API to send requested FD related information from user applications displays to System Viewers (right cursor click).
5. UDS shall provide API's to update the necessary SL-GMS representation of FD's with the current data.
6. UDS shall provide an API to stop updating the SL-GMS display of an FD with the current data.
7. The Data Handler is a collection of FDs used to get the latest values from the CVT. The data handler can get the latest values from the CVT on an update call.
    - 7.1. UDS shall provide the capability to have a SL-GMS container of Data Handlers with a SL-GMS Window State associated with each Data Handler.
        - 7.1.1. UDS shall provide the capability to retrieve a Data Handler from the given SL-GMS Window State.
    - 7.2. UDS shall provide the capability to have a Data Fusion Data Handler.
        - 7.2.1. UDS shall provide the capability to have a list of Data Fused FDs.
        - 7.2.2. UDS shall provide the capability to have a list of FD components with the Fused FD.
        - 7.2.3. UDS shall provide the capability to have a list of fused FDs and their FD components up to the level supported by Data Fusion.

7.2.4. UDS shall provide the capability to display the algorithm/description in string format for a fused FD.

7.3. UDS shall provide the capability to have a Constraint Management Data Handler.

7.3.1. UDS shall provide the capability to get the FD constraint transition with a particular User class.

7.3.2. UDS shall provide the capability to notify Viewers when a constraint transition has been met.

7.3.3. UDS shall provide the capability to get FD constraint transitions in the local Constraint Cache and table to Viewers.

7.3.4. UDS shall provide the capability to give a viewer the FDs constraint expression.

7.3.5. UDS shall provide the capability to give a viewer the FDs constraint transitions.

7.4. UDS shall provide the capability to have a Data Health Data Handler.

7.4.1. UDS shall provide the capability to give the string associated with a health reason code.

7.5. UDS shall provide the capability to have a System/Subsystem Status Data Handler.

7.5.1. UDS shall provide the capability to get a list of FDs associated with System/Subsystem Integrity.

7.5.2. UDS shall provide the capability to retrieve data from the System Configuration Table (SCT).

7.6. UDS shall provide the capability to have a FD Browser Data Handler.

7.6.1. UDS shall provide the capability to get a list of FDs in a user class.

8. FD string formatted functions. FD string formatted functions provide a method to allow the FD value to be returned in a pre-formatted string instead of the raw value.

8.1. UDS shall provide an API to get an Analog FD value back as a formatted string.

8.2. UDS shall provide an API to get a Digital Pattern FD value back as a string.

8.2.1. As a value in Hex.

8.2.2. As a value in Binary.

8.2.3. As a value in Octal.

8.2.4. As a value in BCD.

8.3. UDS shall provide an API to get a Discrete FD value back as a string.

8.3.1. As On/OFF.

8.3.2. As True/False.

8.3.3. As Wet/Dry.

8.3.4. As Open/Close.

8.4. UDS shall provide an API to get an Enumerated FD type back as the string value.

8.5. UDS shall provide an API to get Time FD values back as a string.

8.5.1. For JTOY.

8.5.2. For GMT.

8.5.3. For CDT/MET.

8.5.4. For the day of year.

8.5.5. For the month of year displayed as a number.

8.5.6. For the month of year as a string.

8.5.7. For the day of month.

8.5.8. For the Year.

8.5.9. For the Day of week displayed as a complete string.

8.5.10. For the Day of week displayed as an abbreviated string.

8.5.11. For time as hours.

8.5.12. For time as hours and minutes.

8.5.13. For time as minutes and seconds.

8.5.14. For time as minutes, seconds, and milliseconds.

8.5.15. For time as seconds, and milliseconds.

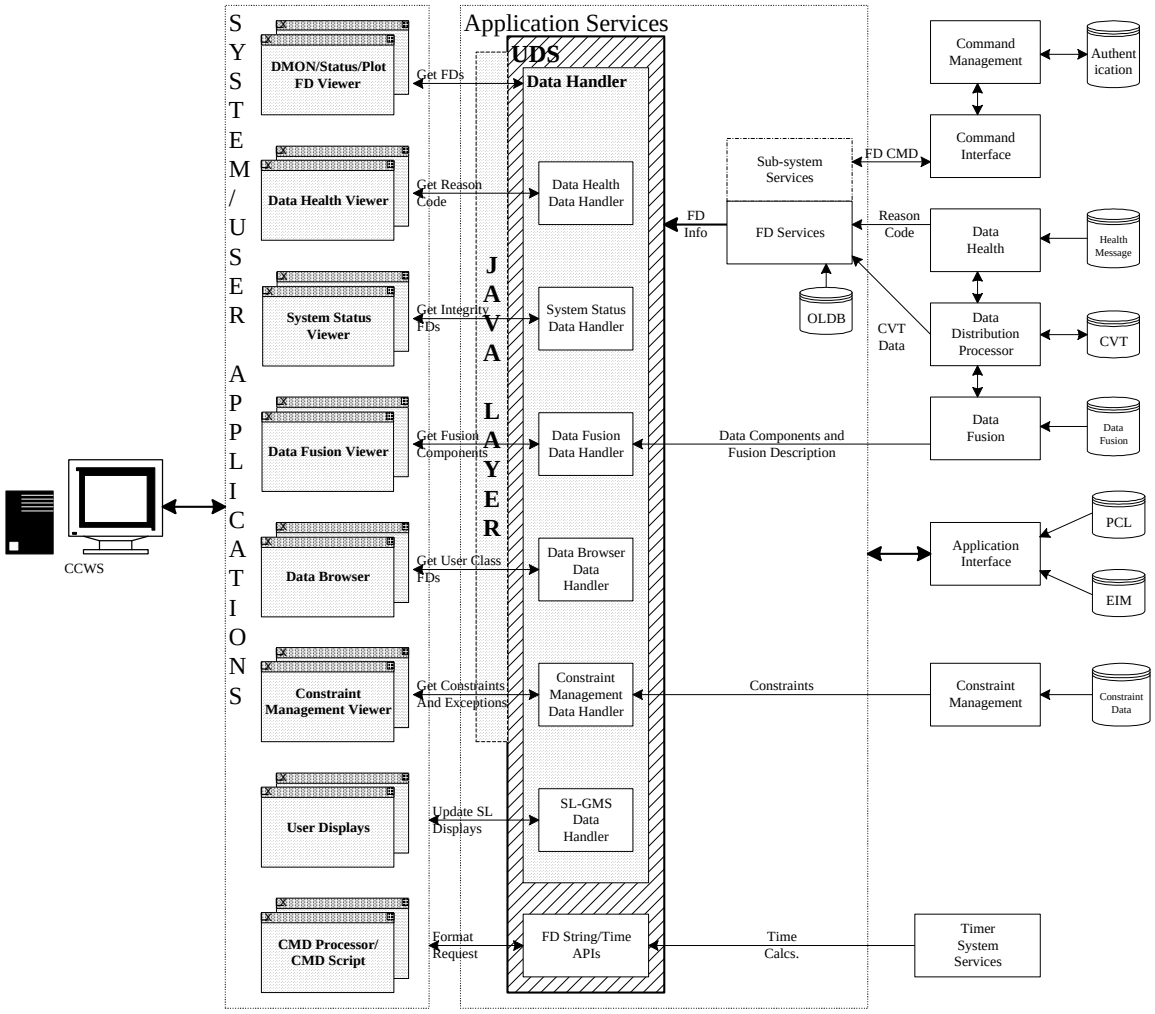
9. UDS shall provide a method to call C++ code from a Java Viewer. This will allow the C++ application and system services to be called from a System Viewer written in Java.

3.2.3 User Display Services Performance Requirements

1. UDS will provide access to updates for the user application display once the latest value is stored in the CVT (UDS performance is based upon data update rates to the local CVT).
2. UDS will provide FD information to System Viewers within one second.

3.2.4 User Display Services Interfaces Data Flow Diagrams

User Display Services Data Flow Diagram



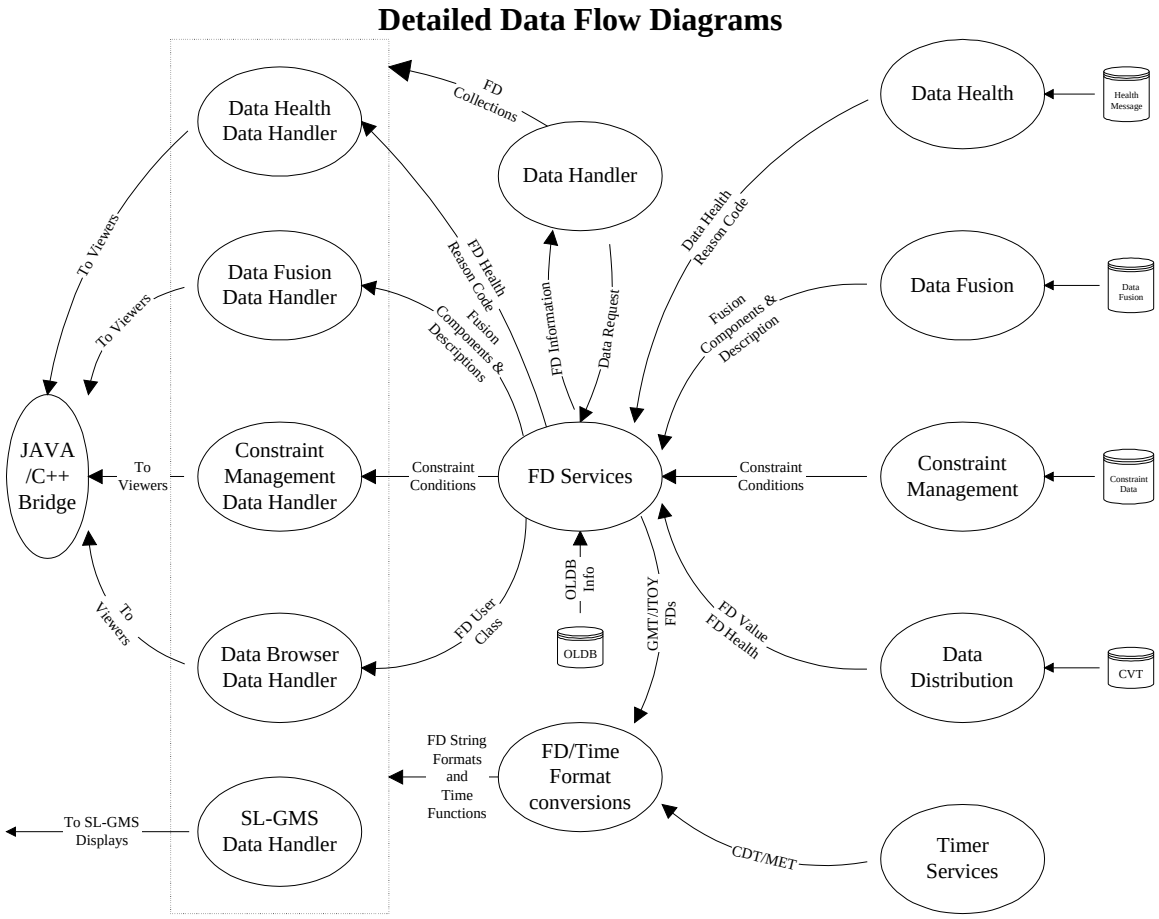
A Command and Control Workstation (CCWS) will contain User Applications, System Viewers, and System Applications. Those applications on the CCWS will go through UDS for Data Handlers, Time APIs, and Message APIs. Each system viewer will have their own special Data Handler that will contain the routines of a base Data Handler, but would also contain any special data necessary for that viewer. System Viewers, since written in Java, will go through a Java interface layer to support the transition to the C++ APIs.

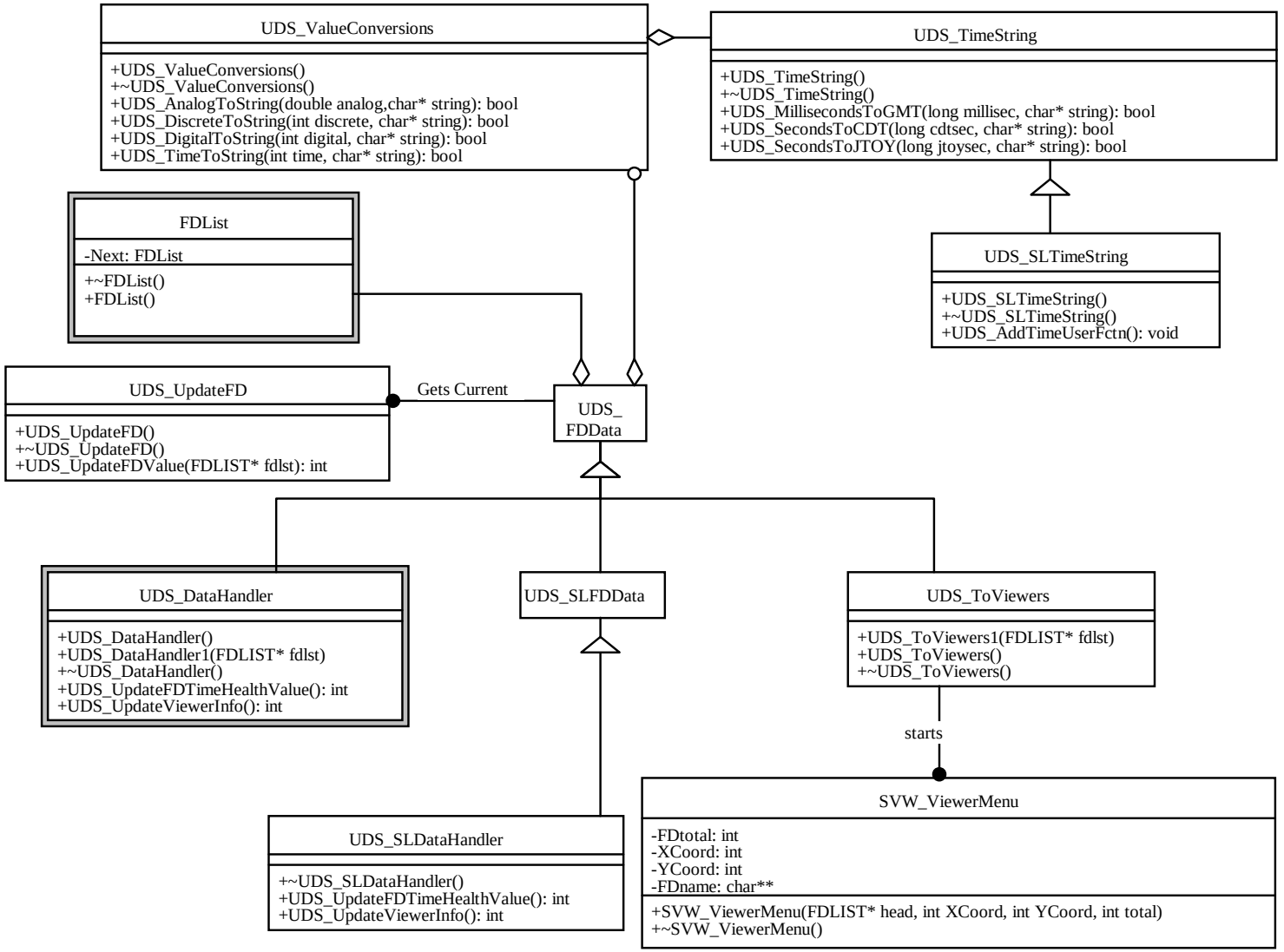
All the Data Handlers will provide access the services provided by FD Services, including Online Data Bank (OLDB) access, CVT access, and Data Health and Fusion access. The Data Handlers will be able to get data regarding Authentication, PCL, EIMs, Constraints, and System Integrity. User Displays will be able to get time conversions displayed on a SL-GMS display with timers, and clocks provided by Timers

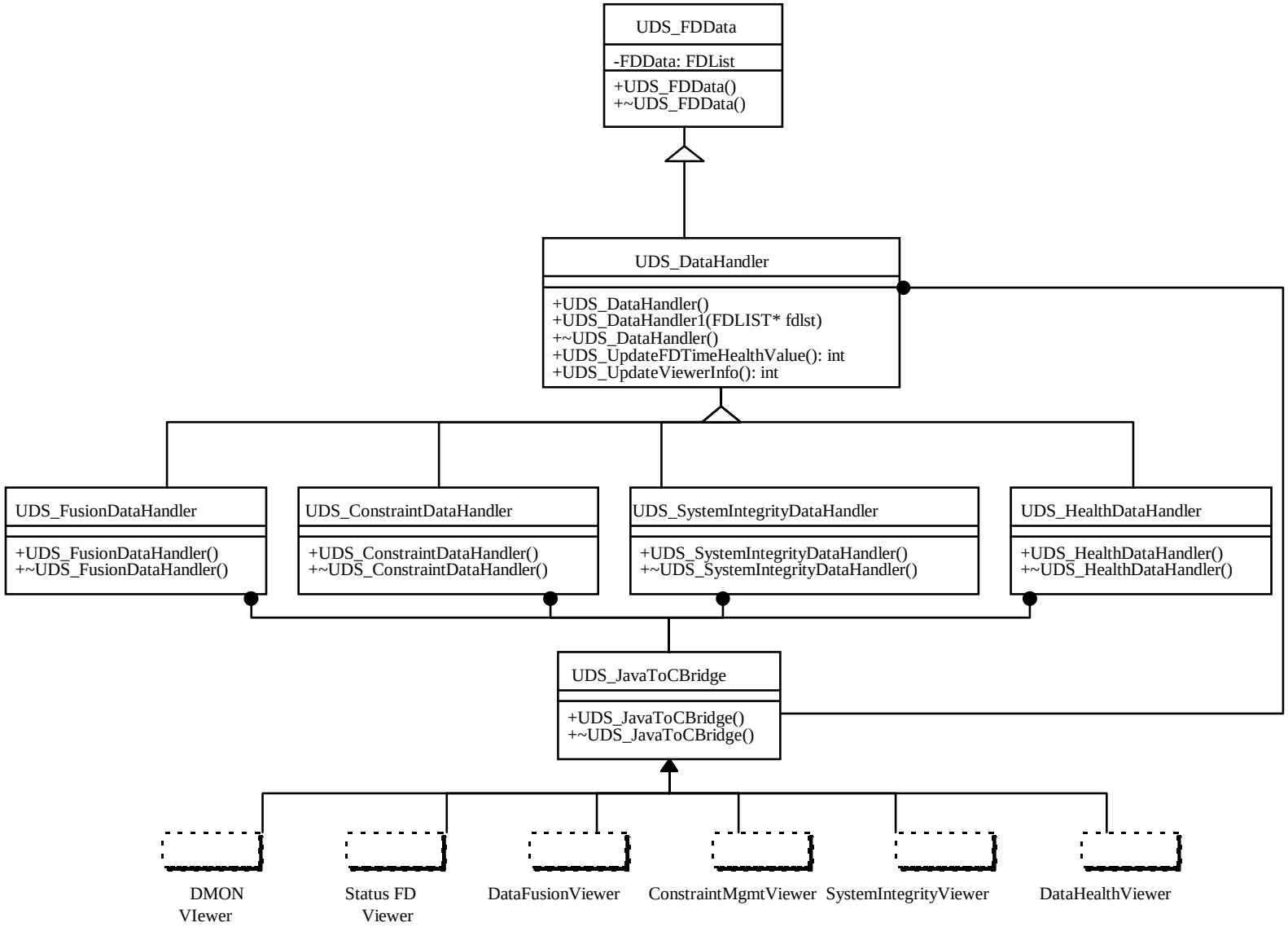
System Services. Command Processor, and Command Scripts will also be able to display a System Message from System Message Services through UDS.

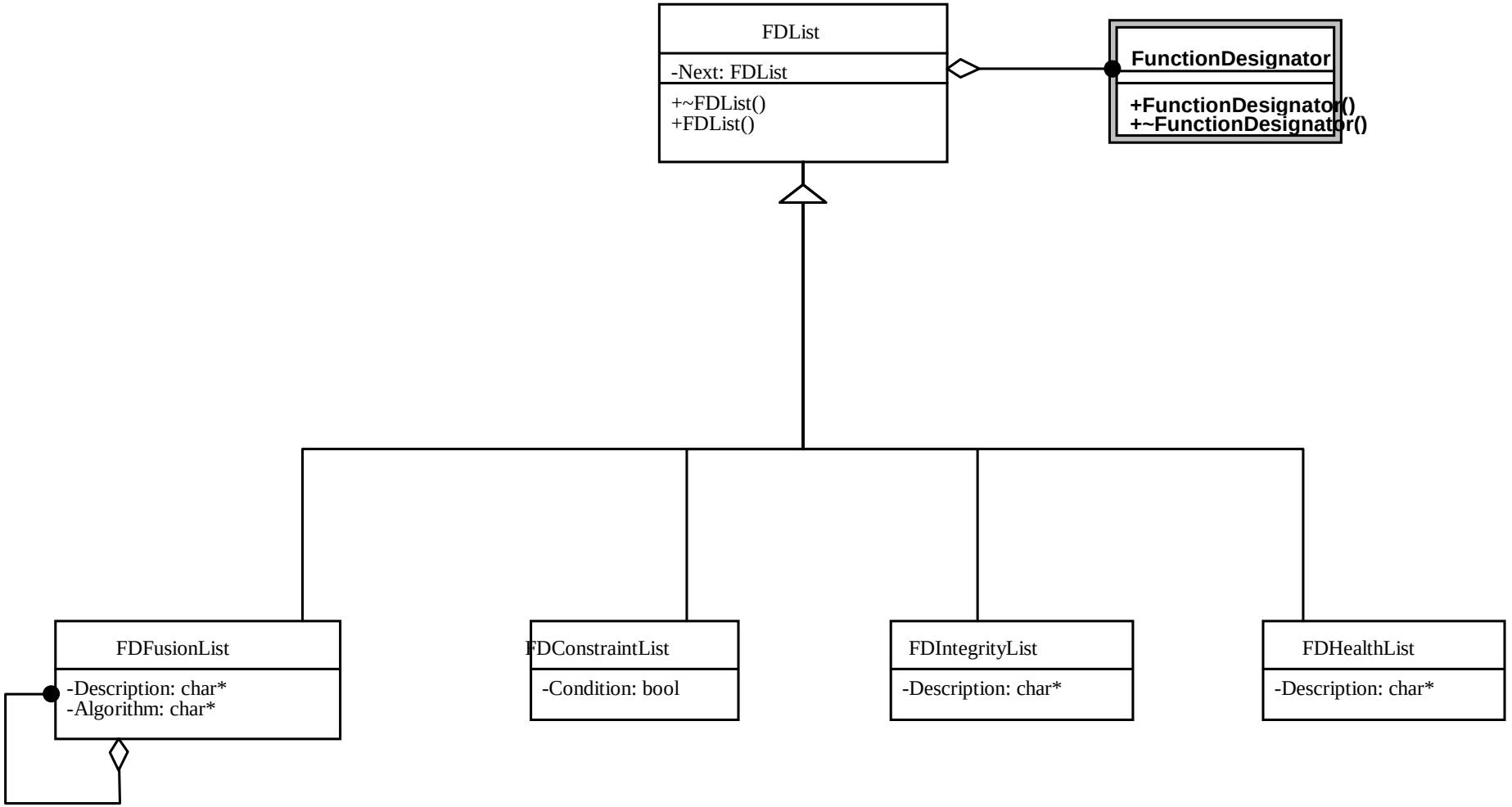
3.3 User Display Services Design Specification

3.3.1 User Display Services Detailed Data Flow




UDS_ValueConversions
+UDS_ValueConversions()
+~UDS_ValueConversions()
+UDS_AnalogToString(double analog,char* string): bool
+UDS_DiscreteToString(int discrete, char* string): bool
+UDS_DigitalToString(int digital, char* string): bool
+UDS_TimeToString(int time, char* string): bool
UDS_TimeString
+UDS_TimeString()
+~UDS_TimeString()
+UDS_MillisecondsToGMT(long millisec, char* string): bool
+UDS_SecondsToCDT(long cdtsec, char* string): bool
+UDS_SecondsToJTOY(long jtoysec, char* string): bool
FDList
-Next: FDList
+~FDList()
+FDList()
UDS_SLTimeString
+UDS_SLTimeString()
+~UDS_SLTimeString()
+UDS_AddTimeUserFctn(): void
UDS_UpdateFD
+UDS_UpdateFD()
+~UDS_UpdateFD()
+UDS_UpdateFDValue(FDLIST* fdlst): int
Gets Current
UDS_
FDData
UDS_DataHandler
+UDS_DataHandler()
+UDS_DataHandler1(FDLIST* fdlst)
+~UDS_DataHandler()
+UDS_UpdateFDTimeHealthValue(): int
+UDS_UpdateViewerInfo(): int
UDS_SLFDData
UDS_ToViewers
+UDS_ToViewers1(FDLIST* fdlst)
+UDS_ToViewers()
+~UDS_ToViewers()
starts
SVW_ViewerMenu
-FDtotal: int
-XCoord: int
-YCoord: int
-FDname: char**
+SVW_ViewerMenu(FDLIST* head, int XCoord, int YCoord, int total)
+~SVW_ViewerMenu()
UDS_SLDataHandler
+~UDS_SLDataHandler()
+UDS_UpdateFDTimeHealthValue(): int
+UDS_UpdateViewerInfo(): int


UDS_FDData
-FDData: FDList
+UDS_FDData()
+~UDS_FDData()
UDS_DataHandler
+UDS_DataHandler()
+UDS_DataHandler1(FDLIST* fdlst)
+~UDS_DataHandler()
+UDS_UpdateFDTimeHealthValue(): int
+UDS_UpdateViewerInfo(): int
UDS_FusionDataHandler
+UDS_FusionDataHandler()
+~UDS_FusionDataHandler()
UDS_ConstraintDataHandler
+UDS_ConstraintDataHandler()
+~UDS_ConstraintDataHandler()
UDS_SystemIntegrityDataHandler
+UDS_SystemIntegrityDataHandler()
+~UDS_SystemIntegrityDataHandler()
UDS_HealthDataHandler
+UDS_HealthDataHandler()
+~UDS_HealthDataHandler()
UDS_JavaToCBridge
+UDS_JavaToCBridge()
+~UDS_JavaToCBridge()
DMON VIewer
Status FD Viewer
DataFusionViewer
ConstraintMgmtViewer
SystemIntegrityViewer
DataHealthViewer


FDList
-Next: FDList
+~FDList()
+FDList()
FunctionDesignator
+FunctionDesignator()
+~FunctionDesignator()
FDFusionList
-Description: char*
-Algorithm: char*
FDConstraintList
-Condition: bool
FDIntegrityList
-Description: char*
FDHealthList
-Description: char*

3.3.2 User Display Services External Interfaces

3.3.2.1 User Display Services Message Formats

User Display Services outputs the following system messages:

Message Number = ASV_UDS_NOSYSVWR
Message Group = ASV
Severity = Error

System Viewers #ARGUMENT1# could not be found.

ARGUMENT1 = ASCII character string representing the filename and location of the System Viewer executable.

Help Information Content

The environment variable $ASV_SVW_EXEC must be defined. This variable must point to the location of the System Viewers executable.

Detailed Information
N/A

3.3.2.2 UDS Display Formats

Not applicable. UDS does not provide any displays.

3.3.2.3 UD Services Input Formats

Not applicable. There are no language-like interfaces provided by UDS.

3.3.2.4 Recorded Data

UDS does not record data nor initiate data recording.

3.3.2.5 UDS Printer Formats

UDS does not provide printed information.

3.3.2.6 Inter-process Communications

Not applicable.

3.3.2.7 UDS External Interface Calls

The CLCS UDS Interface Description Document #84K00361 describes the data sent between the UDS CSC and CLCS applications via a calling mechanism. The CLCS FD Services Interface Description Document #84K00362 describes the data sent between UDS and FD Services.

3.3.2.8 UDS Table Formats

UDS does not utilize any tables internally that are provided from an outside source.

3.3.3 UDS Test Plan

3.3.3.1 UDS Data Handler for SL-GMS User Displays

Test Objective:
Test and / or Regression test the following User Display Services (UDS) capabilities to output FD and OLDB data to SL-GMS-based displays:

1. Display FD (CVT) values on an SL-GMS display at a CCW/S: (5.2.2.1)
    - 1.1. Analog (one or more of each analog data type - TBD data types)
    - 1.2. Discrete (one or more of each discrete data type - 8 data types)
    - 1.3. Digital Pattern (one or more of each of the following data types: Hex, Octal, BCD, Enumerated Type)
    - 1.4. Time values in the following formats (JTOY, CDT/MET, others?)
2. Allow user application displays access (to) FD related information from the CVT/OLDB. (5.2.2.2)
3. Update the necessary SL-GMS representation of FD's with the current data (5.2.2.5).
4. Stop updating the SL-GMS display of an FD with the current data (5.2.2.6).
5. Verify each API call returns correct status (success only).

Test Approach Summary:
The test conductor will bring up an SL-GMS test display containing one (or more) FD of each data type with the value, time, health, and OLDB parameters. In another window, CVT_Look or another test tool will be used to examine the current value for each FD on the test display. The values will be compared. Next, the Sim Gateway will be started, and FD value changes will be verified by stopping the Sim G/W and comparing the CVT values to the values on the test display. The Sim G/W will be restarted, and the Data Handler will be set to stop updating values. The CVT will be examined to show values are being changed in the CVT, but not on the test display. The Data Handler will be set to start updating the display again, and the display will be compared to the CVT to show the values are being updated correctly.
Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor TCID
- Data Distribution / CVT on DDP, CCP, and CCWS
- **SL-GMS-based test display**
- TBD CVT Viewing tool for CCWS to examine CVT values (CVT_LOOK from DD or other tool)

Requirement(s):

- SLS Requirement(s): 5.2.5.8.3
- ASV/UDS Requirement(s): 5.2.2.1, 5.2.2.2, 5.2.2.5, 5.2.2.6
- RTC Applications (User) Requirement(s): N/A

3.3.3.2 UDS Data Handler for Data Fusion

Test Objective:

Test and / or Regression test the following UDS capabilities to output Fusion FD value and parameter data to SL-GMS displays:

1. Provide a list of Fused FDs at a CCW/S: (5.2.2.7.2.1)
    1.1. Analog (one or more of each analog data type - TBD data types)
    1.2. Discrete (one or more of each discrete data type - 8 data types)
    1.3. Digital Pattern (one or more of each of the following data types: Hex, Octal, BCD, Enumerated Type)
2. Provide a list of FD components with the Fused FD. (5.2.2.7.2.2)
3. Provide a list of fused FDs and their FD components up to the level supported by Data Fusion. (5.2.2.7.2.3)
4. Display the algorithm/description in string format for a fused FD. (5.2.2.7.2.4)
5. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

1. The test conductor will use the Fusion FD Viewer or a test display to view the output of one or more Fusion FDs. The Fusion Algorithm Viewer will be used to view the equation. The FD Viewer or Browser will be used to view the input values. Using static or dynamic data, the Fusion FD will be executed, and the output will be compared to the expected result, using the values of the input FD(s) and the equation.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS
- **Data Fusion**
- **Data Fusion Viewer** (to view and control Fusion execution)
- **Data Fusion Algorithm Viewer**
- **Command Management** or equivalent mechanism to send Writes to Gateways

Requirement(s):

- SLS Requirement(s): ?
- FD Services Requirement(s): 5.2.2.7.2.1, 5.2.2.7.2.2, 5.2.2.7.2.3, 5.2.2.7.2.4
- RTC Applications (User) Requirement(s): 4.2.4.2.1, 4.2.4.2.2, 4.2.4.2.3

3.3.3.3 UDS Data Handler for Constraint Management

Test Objective:

Test and / or Regression test the following UDS capabilities to provide constraint data:

1. Get the FD constraint transition for a particular user class. (5.2.2.7.3.1)
2. Notify Viewers when a constraint transition has been met. (5.2.2.7.3.2)
3. Get FD constraint transitions in the local Constraint Cache and table to Viewers. (5.2.2.7.3.3)
4. Give a viewer the FD's constraint expression. (5.2.2.7.3.4)
5. Give a viewer the FD's constraint transitions. (5.2.2.7.3.5)
6. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

The test conductor at a CCWS will use a Constraint Management-provided viewer to set constraints for specified FDs of each FD type. A viewer will be used to verify the constraint data was received correctly, that the constraint was violated (transition from in limits to out of limits) and that the constraint returned to normal (transition from out of limits to in limits). A constraint expression/algorithm viewer will be used to view a constraint expression.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor test TCID, with Fusion FDs and pseudo FDs defined

- Thor test Constraint Algorithm table, provided by DB Safe personnel.
- Data Distribution / CVT on DDP, CCP, and CCWS
- **Constraint Management** or equivalent mechanism to set Constraints for FDs and provide constraint transition status
- Command Processor or equivalent to input FD writes at CCWS
- TBD Constraint Viewing tool for CCWS to examine Constraint values

Requirement(s):

- SLS Requirement(s): 2.2.5.4.1
- ASV / UDS Requirement(s): 5.2.2.7.2.1, 5.2.2.7.2.2, 5.2.2.7.2.3, 5.2.2.7.2.4.
- RTC Applications (User) Requirement(s): N/A.

3.3.3.4 UDS Data Handler for Data Health

Test Objective:

Test and / or Regression test the following UDS Data Health capabilities:

1. Give the string associated with a health reason code (5.2.2.7.4.1)
2. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

The test conductor will bring up the Data Health viewer or a test display and a TBD viewer or test display to set health status. For one or more FDs, the health reason code will be set and compared to the expected value.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS
- **Command Management** or equivalent mechanism to send Writes to Gateways
- Command Processor or equivalent to write FD Health status at CCWS
- **Data Health**
- **TBD Viewing tool for CCWS to examine Data Health values**

Requirement(s):

- SLS Requirement(s):
- FD Services Requirement(s): 5.2.2.7.4.1
- RTC Applications (User) Requirement(s): 4.2.4.4.1, 4.2.4.4.2.1-7???

3.3.3.5 UDS Data Handler for System / Subsystem Status

Test Objective:

Test the following UDS capabilities for System / Subsystem Integrity:

1. Get a list of FDs associated with System/Subsystem Integrity. (5.2.2.7.5.1)
2. Retrieve data from the System Configuration Table (SCT). (5.2.2.7.5.2)
3. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

At a CCWS, the test conductor will use the System/Subsystem Viewer or an equivalent test display to retrieve a pre-defined list of FDs associated with System/Subsystem Integrity. This will be compared to a printout of the pre-defined list. Next, the test conductor will read the SCT, and display the data on a TBD Viewer or test display. This will be compared to a printout of the pre-defined table data.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor TCID, **with System/Sub-system Integrity FDs** defined

- Data Distribution / CVT on DDP, CCP, and CCWS
- **System / Subsystem Integrity or equivalent test tool**
- **TBD Viewing tool or test displays to examine System / Subsystem Integrity FD data and SCT data**

Requirement(s):

- SLS Requirement(s):
- FD Services Requirement(s): 5.2.2.7.5.1, 5.2.2.7.5.2
- RTC Applications (User) Requirement(s):

3.3.3.6 UDS Data Handler for FD Browser

Test Objective:

Test the following FD Svcs capabilities to write FD value and parameter data:

1. Get a list of FDs in a user class. (5.2.2.7.6.1)
    - 1.1. Analog (one or more of each analog data type - TBD data types)
    - 1.2. Discrete (one or more of each discrete data type - 8 data types)
    - 1.3. Digital Pattern (one or more of each of the following data types: Hex, Octal, BCD, Enumerated Type)
    - 1.4. Time values in the following formats (JTOY, CDT/MET, others?)
2. Verify each API call returns correct status (success only).

Test Approach Summary:

The test conductor will create a list of FDs with a defined user class, save it, and retrieve (use) it from the FD Browser. The FDs displayed on the Browser will be compared to the list that was created.

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN and DCN
- Reliable Messaging / Network Services
- Thor TCID, **with User Classes defined**
- Data Distribution / CVT on DDP, CCP, and CCWS
- **FD Browser**
- Command Management or equivalent to write FD list at CCWS

Requirement(s):

- SLS Requirement(s):
- FD Services Requirement(s): 5.2.2.7.6.1
- RTC Applications (User) Requirement(s):

3.3.3.7 UDS String Formatting Functions

Test Objective:

Test the following UDS capabilities for formatting data into strings:

1. Get an analog FD value back as a formatted string (5.2.2.8.1)
2. Get a digital pattern FD value back as a string for the following data types: (5.2.2.8.2)
    - 2.1. hex
    - 2.2. binary
    - 2.3. octal
    - 2.4. BCD
3. Get a discrete FD value back as a string. (5.2.2.8.3)
    - 3.1. On / Off
    - 3.2. True / False
    - 3.3. Wet / Dry
    - 3.4. Open / Closed
4. Get an enumerated FD value back as a string. (5.2.2.8.4)
5. Get Time FD values back as a string: (5.2.2.8.5)
    - 5.1. JTOY
    - 5.2. GMT

5.3. CDT/MET
5.4. day of year
5.5. month of year, displayed as a number
5.6. month of year as a string
5.7. day of month
5.8. Year
5.9. day of week displayed as a complete string
5.10. day of week displayed as an abbreviated string
5.11. time as hours
5.12. time as hours and minutes
5.13. time as minutes and seconds
5.14. time as minutes, seconds, and milliseconds
5.15. time as seconds and milliseconds

6. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

At a CCWS, the test conductor will use a viewer or test display with the time formats built in string formats and a viewer or test display showing the times in raw format. A comparison will be made by using a calculator to convert the raw times into processed, string formats. The time values in the CVT can be dynamic (using Gateway or Sim Gateway data) or static (using pre-defined values, manually placed into the CVT using CVT_stuffer or some other tool).

Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN (100BaseT) and DCN (FDDI)
- Reliable Messaging (or equivalent) / Network Services
- Thor TCID, with Time FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS **OR** CVT_stuffer tool
- **System Services' Timer Services**
- **TBD Viewing tool for CCWS to examine time values as strings OR test display**
- **TBD Viewing tool for CCWS to examine raw time values OR test display**

Requirement(s):

- SLS Requirement(s):
- FD Services Requirement(s): 5.2.2.8.1, 5.2.2.8.2, 5.2.2.8.3, 5.2.2.8.4, 5.2.2.8.5
- RTC Applications (User) Requirement(s):

3.3.3.8 UDS Java Interface to System Viewers

Test Objective:

1. Test the following UDS capability to call C++ code from a Java Viewer, such that C++ application and system services can be called from a System Viewer (written in Java). (5.2.2.9)
2. Verify each API call returns correct status (success only). (1.2.2.1)

Test Approach Summary:

The test conductor will bring up each Java-based viewer and demonstrate the browser receives data correctly.

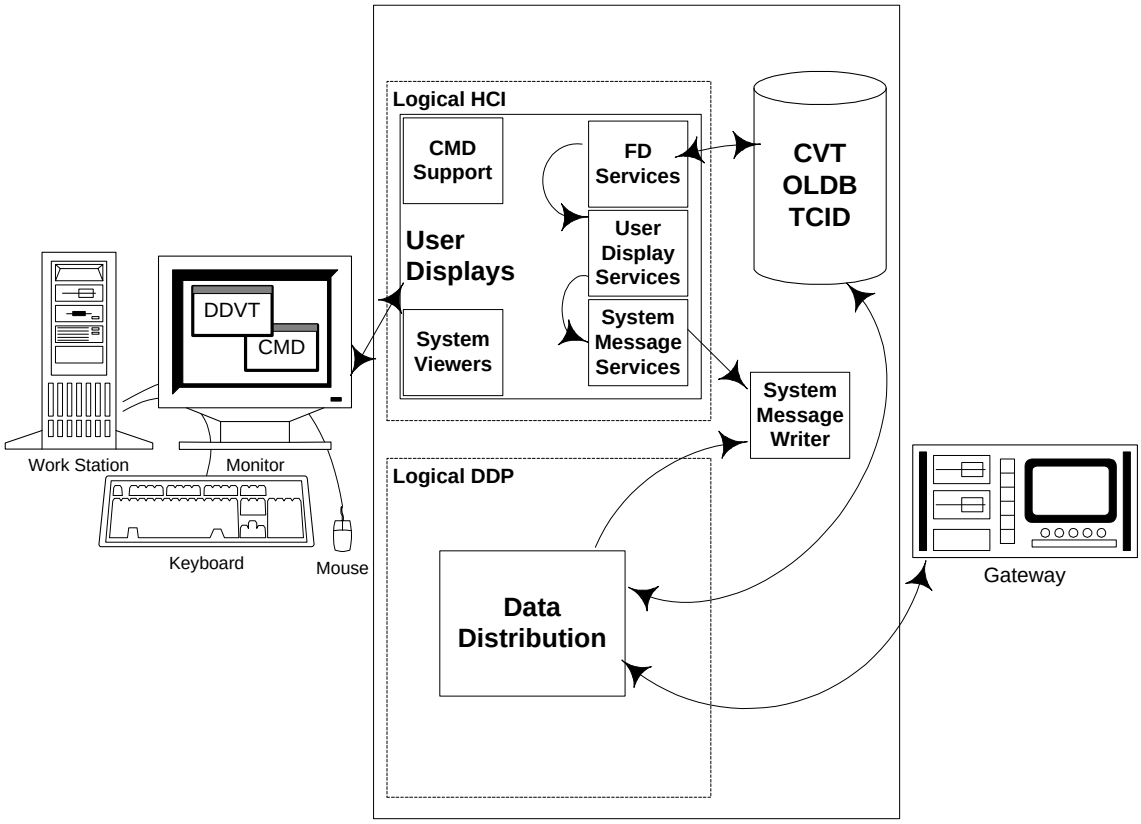
Required Test Configuration / Dependencies:

- Ops CM to Configure Flow Zone
- DDP, CCP, and CCWS platforms
- RTCN and DCN
- Reliable Messaging / Network Services
- Thor TCID, with Fusion FDs and pseudo FDs defined
- Data Distribution / CVT on DDP, CCP, and CCWS
- TBD CVT Viewing tool for CCWS to examine CVT values (CVT_LOOK from DD or other tool)

Requirement(s):

- SLS Requirement(s):
- FD Services Requirement(s):): 5.2.2.9

- RTC Applications (User) Requirement(s):



UDS Minimum Test Configuration

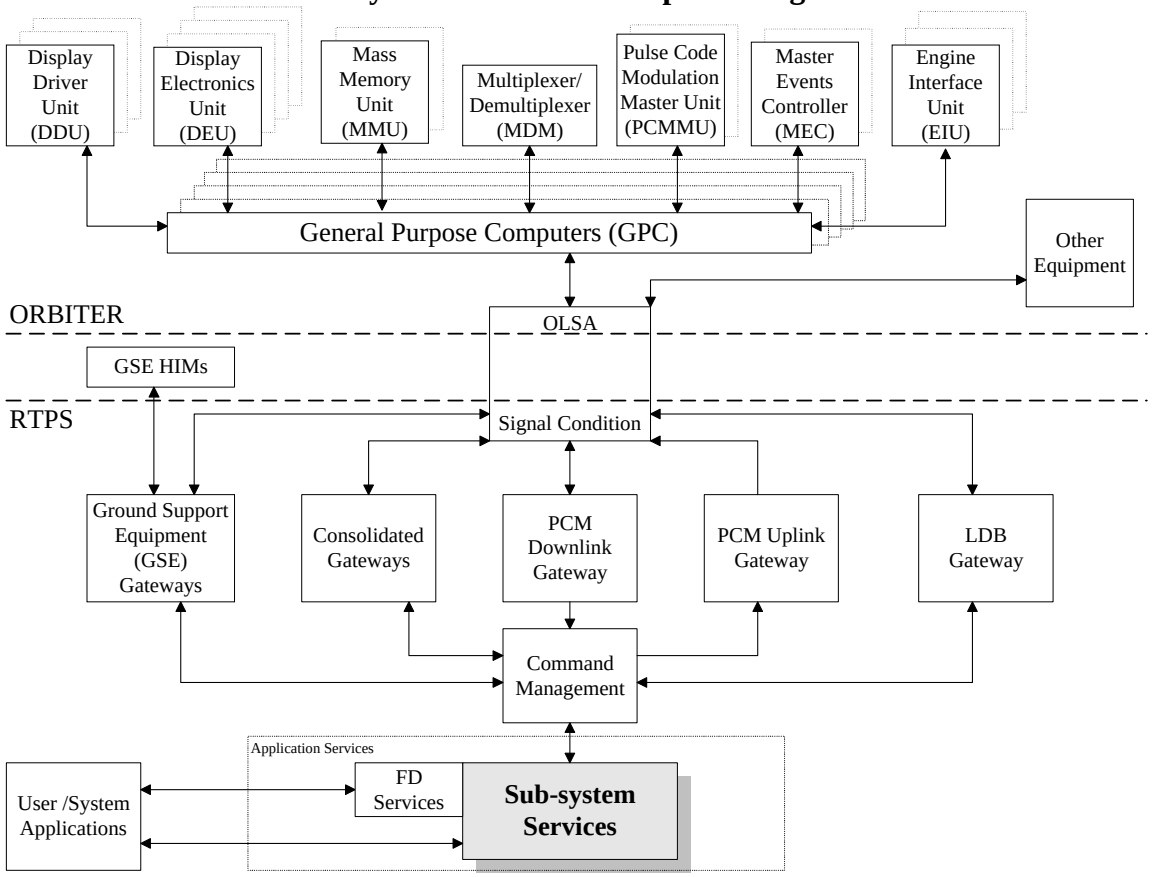
4. Sub-system Services (SSS) CSC

4.1 Sub-system Services Introduction

4.1.1 Sub-system Services Overview

Sub-system Services (SSS) is a CSC in Application Services (ASV). SSS consist of three parts, Gateway Interface Services (GIS), On-board Services (OBS), and Non-gateway Services (NGS). GIS will be the interface for user applications to command Function Designators (FDs) and non-FDs to/from Ground Support Equipment (GSE) and Pulse Code Modulation (PCM) Down-link gateways. OBS will interface the commands for user applications for FDs and non-FDs from the Launch Data Bus (LDB) and PCM up-link gateways. Non-gateway Services will interface with commands that have a destination not associated with a gateway. This includes setting time FDs, health of an FD, and pseudo FDs.

Sub-system Services Conceptual Diagram



4.1.2 Sub-system Services Operational Description

Sub-system Services (SSS) will be called from user and system applications. SSS will provide type checking C++ Application Programmers Interfaces (APIs) so that commands from an application are checked at compile and link time instead of run time. SSS commands will consist of two parts. FD related commands, and non-FD related commands. An FD Services object inherits FD commands from a sub-system class specific to the commands destination. This will provide data hiding and allow the object to only call a command that the FD is capable. Non-FD commands will be able to be accessed through SSS directly, but will not include any commands associated with a FD. The sub-system command will have in the object the necessary information about the particular sub-system. This will provide the syntax checking necessary to ensure the correct command. SSS will then put all the necessary parts of the command

together to be sent out to Command Management. Command responses will then be returned to SSS to provide the information back to the FD, or to the application.

4.2 Sub-system Services Specifications

4.2.1 Sub-system Services Groundrules

- Command Management will provide all gateway communications and packet definition for Sub-system Services.
- User applications must use SSS to send commands, or use FD Services which will access SSS For FD Commands.
- Sub-system Services will only read or command Onboard FDs for MDM in Thor.

4.2.2 Sub-system Services Functional Requirements

4.2.2.1 On-board Services (OBS) Functional Requirements

1. OBS shall provide a method for building a raw data word for discrete, analog, and digital pattern measurement and stimulus data (KSC-LPS-OP-033-4 section 2.1 (TRANSLATE TO RAW DATA AND MERGE WITH)).[4]

2. OBS shall provide a method for converting a raw data word from onboard memory into measurement data (KSC-LPS-OP-033-4 section 2.2 (CONVERT RAW DATA)).

3. OBS shall provide functionality for an user application to issue quantity type data to the system under test (KSC-LPS-OP-033-4 section 3.1 (APPLY ANALOG)).

   3.1. Multiplexer/Demultiplexer (MDM).

   3.2. FLEX MDM.

   3.3. Engine Interface Unit (EIU).

   3.4. Sequence Control Assembly (SCA).

   3.5. Master Events Controller for critical and critical/non-critical operations.

   3.6. MDM or FLEX MDM PROM sequence.

   3.7. For a single analog command applied to a single analog FD.

   3.8. For a single analog command applied to many analog FDs. [5]

   3.9. For many analog commands applied to many analog FDs.

4. OBS shall provide functionality for an user application the ability to command an Engine Interface Unit (EIU) to perform the internal functions (KSC-LPS-OP-033-4 section 3.2.1):

   4.1. Status Override (COMMAND EIU STATUS OVERRIDE).

   4.2. Master Reset (COMMAND EIU MASTER RESET).

   4.3. Wrap Test (COMMAND EIU STATUS OVERRIDE).

5. OBS shall provide functionality to send a request to the Launch Sequence Functional Destination for controlling the terminal count to support (COMMAND LAUNCH SEQUENCE (KSC-LPS-OP-033-4 section 3.2.2)):

   5.1. RS Auto Sequence Start (RS AUTO SEQUENCE START).

   5.2. Hold (HOLD).

   5.3. Recycle (RECYCLE).

   5.4. Bypass of LO2 Overboard Bleed Valve CL A (BYPASS OF LO2 OVBD BLEED VLV CL A).

---

[4] Converting raw data to engineering units will be done at the gateway.

[5] The functionality to give bundled commands will probably be implemented in higher level software such as Control-Shell, or a data handler.

5.5. Bypass of LO2 Overboard Bleed Valve CL B (BYPASS OF LO2 OVBD BLEED VLV CL B).
5.6. Bypass of LO2 Accumulator Re-circulation Valve OP (BYPASS OF LO2 ACCUM RECIRC VLV OP).
5.7. Resume Count (RESUME COUNT).
5.8. SRB FCS Hydraulic Verification Flag (SRB FCS HYDR VERIFICATION FLAG).
5.9. Orbiter Vent Doors Override (ORBITER VENT DOORS OVERRIDE).
5.10. Estimated Mass of Orbiter with External Tank (EST MASS OF ORBITER WITH ET).
5.11. Aerosurface Drive Check (AEROSURFACE DRIVE CHECK).
5.12. MPS/ET Low Level Sensor Disable Flag (MPS/ET LOW LVL SNCR DSBL WRD).
5.13. LPS Go For Engine Start (LPS GO FOR ENGINE START).
5.14. ET LH2 Low Level Sensor Disable Flag (ET LH2 LOW LVL SNCR DSBL FLAG).
5.15. SRM Chamber Pressure Calibration Word (SRM CHAMBER PRESS CAL WRD).
5.16. *JTOY of Lift Off To (JTOY OF LIFT OFF TO).*

6. OBS shall provide functionality to request an MDM or *FLEX MDM* to perform the internal functions (KSC-LPS-OP033-4 section 3.2.3):
   6.1. Master Reset (COMMAND MDM MASTER RESET).
   6.2. Perform BITE Tests 1, 2, 3, and 4 (COMMAND MDM BITE).
   6.3. Load the BITE Status Register (COMMAND MDM BITE STATUS REGISTER).
   6.4. Perform a Wrap Test (COMMAND MDM WRAP TEST).
7. OBS shall provide the functionality to communicate with the Master Event Controller BTUs and to command their test and control functions for (KSC-LPS-OP-033-4 section 3.2.4):
   7.1. SIM PIC CAP Volt Enable (COMMAND MEC SIM PIC CAP VOLT).
   7.2. Master Reset (COMMAND MEC MASTER RESET).
   7.3. Wrap Test (COMMAND MEC WRAP TEST).
8. OBS shall provide functionality to request the Sequence Control Assembly (SCA) to perform internal test functions to (KSC-LPS-OP-033-4 section 3.2.5):
   8.1. Wrap Test (COMMAND SCA WRAP TEST).
   8.2. BITE Status Register (COMMAND SCA BITE STATUS REGISTER).
9. OBS shall provide functionality to request the up-link gateway to perform the specified functions (KSC-LPS-OP-033-4 section 3.2.6):
   9.1. to execute an up-link request which has been loaded in the two-stage buffer by a previous statement (COMMAND UPLINK TWO STAGE BUFFER EXECUTE).
   9.2. to clear a request in the two-stage buffer previously loaded by a procedure (COMMAND UPLINK TWO STAGE BUFFER CLEAR).
   9.3. to request the up-link gateway to accept a 48 bit pre-formatted command data word and cause it to be issued to the onboard system (COMMAND UPLINK TO ISSUE).
10. OBS shall provide functionality to provide an issue statement to support commands to the on-board components (KSC-LPS-OP-033-4 section 3.2.7 (ISSUE)):
   10.1. Multiplexer/Demultiplexer (MDM)
   10.2. FLEX MDM.
   10.3. MDM Serial IO Device.
   10.4. Engine Interface Unit (EIU)
   10.5. Sequence Control Assembly (SCA)
   10.6. Maser Events Controller for critical and critical/non-critical operations.
   10.7. *MDM or FLEX MDM PROM sequence.*
11. OBS shall provide functionality to provide “read” operations on the on-board components (KSC-LPS-OP-033-4 section 3.3):
   11.1. Engine Interface Unit (READ EIU)
   11.2. Multiplexer/Demultiplexer (READ MDM)
   11.3. Master Events Controler (READ MEC)
   11.4. Pulse Code Modulation Master Unit (READ PCMMU).
   11.5. Sequence Control Assembly (READ SCA).
   11.6. Support a single FD to be read and stored in a single variable.

11.7. Support a single FD to be read and stored in multiple variables. [6]

11.8. *Support multiple FDs to be read and stored in multiple variables.*

12. OBS shall provide the capability to set discrete statements for MDMs, *MEC, and FLEX MDMs* for (KSC-LPS-OP-033-4 section 3.4 (SET)):

12.1. Support time intervals to issue a discrete command and issue the complement when the time value specified has expired.

12.2. Support time interval to reissue the same state after the time interval has been exhausted (NO COMPLEMENT).

12.3. Support commands to receive high priority processing by the LDB gateway and routing to the SACS functional destination with responses inhibited (CRITICAL).

12.4. Support the up-link gateway to command data to be issued via the 2-stage buffer (MDM MULTIPLE).

12.5. Support up-link gateway to cause a command data to be issued via the stored program command buffer and specify the JTOY at which the data is to be issued (AT TIME VALUE).

12.6. Support to request the PCM up-link gateway to format a MDM single command for repeated issuance at the current system rate, and inhibit all other command processing until a STOP command is issued (REPEATED).

12.7. A single discrete command applied to a single FD.

12.8. A single discrete command applied to many FDs.

12.9. *Many discrete commands applied to many FDs*.

13. OBS shall provide the functionality to cancel or terminate the execution of an on-board explicitly coded program (ECP) or TCS sequence (KSC-LPS-OP-033-4 section 4.1).

13.1. Cancel the execution of an ECP (CANCEL ECP PROGRAM).

13.2. Cancel the execution of a TCS sequence (CANCEL TCS PROGRAM).

14. OBS shall provide the functionality to initiate the parallel execution of a TCS sequence or ECP in the GPC which is currently communicating with the ground via the LDB and support (KSC-LPS-OP-033-4 section 4.2 (CONCURRENT)):

14.1. TCS sequence selected from mass memory and will continue as soon as the executor receives notification from the LDB gateway that the last data block of the sequence has been transferred to the GPC (TCS SEQUENCE FROM MASS MEMORY).

15. OBS shall provide the functionality to initiate the execution of TCS Sequence (KSC-LPS-OP-033-4 section 4.3 (PERFORM TCS PROGRAM)).

16. OBS shall provide the functionality to restart execution of a previously stopped TCS sequence with an option to specify which TCS step number is to be used to re-start execution (KSC-LPS-OP-033-4 section 4.4 (RESUME TCS PROGRAM)).

17. OBS shall provide the capability to temporarily halt the execution of a TCS sequence, to stop the execution of a repeated MDM single command in the PCM up-link gateway or to stop the execution of a repeated payload throughput command in the PCM up-link gateway (KSC-LPS-OP-033-4.section 4.5 (STOP TCS PROGRAM)).

18. OBS shall provide the functionality to issue statements to SRB MDM's to (KSC-LPS-OP-033-4 section 6.0):

18.1. lock the MDM module specified in the data bank for the given FD (LOCK SRB MDM).

18.2. unlock the MDM module specified in the data bank for the given FD (UNLOCK SRB MDM).

19. *OBS shall provide the functionality to request the LDB gateway to enable the currently inactive LDB to provide the capability to switch LDBs when GPCs are polling simultaneously on both LDBs (KSC-LPSOP-033-4 section 7.0 (SWITCH LDB)).*

20. OBS shall provide the functionality to provide the capability to control the LDB I/O functions performed by the GPC. This capability shall cause the desired request to be sent to the GPC which is currently communicating with the LDB gateway so that the current mode and/or control paths are changed by the GPC (KSC-LPS-OP-033-4 section 8.1 (LDB CONTROL)).

---

[6] The functionality to support multiple FD handling may be implemented in a higher level such as Control-Shell or a data handler.

21. OBS shall provide the functionality to obtain the contents of on-board memory and store the data for (KSC-LPS-OP-033-4 section 8.2):
    21.1. GPC memory (READ GMEM) (contiguous reads only for Thor).
    21.2. DEU memory (READ DEU MEM).
    21.3. SSME memory (READ SSME MEM).
22. OBS shall provide the functionality to modify locations in the on-board memory for (KSC-LPS-OP-033-4 section 8.3):
    22.1. GPC memory (WRITE GMEM) (contiguous writes only for Thor).
    22.2. SSME memory (WRITE SSME MEM).
23. OBS shall provide the functionality to modify the value of data contained in the TCS 1-1 registers in the GPC numbered 49 through 96 without being required to specify the register's address (KSC-LPS-OP-033-4 section 9 (LOAD REGISTER)).
24. OBS shall provide the functionality to send values in the TCS 1-1 registers in the GPC numbered 49 through 96 to the ground without being required to specify the register's address (KSC-LPS-OP-033-4 section 10 (DUMP REGISTER)).
25. OBS shall provide the functionality to emulate the DEU keystrokes and display data to the onboard CRT
    25.1. Provide functionality to specify the value of a DEU type command in any of the permissible formats to the on-board system under test (KSC-LPS-OP-033-4 section 11.1 (ISSUE DEU)).
        25.1.1. Operational Sequence (OPS).
        25.1.2. Specialist Function/Display Function (SPEC).
        25.1.3. Item (ITEM).
        25.1.4. Resume (RESUME).
        25.1.5. Proceed (PRO).
        25.1.6. Execute (EXEC).
        25.1.7. Message Reset (MSG RESET).
        25.1.8. Acknowledge (ACK).
        25.1.9. Fault Summary (FAULT SUMM)
        25.1.10. Computer/CRT (GPC/CRT).
        25.1.11. System Summary (SYS SUMM).
        25.1.12. I/O Reset (I/O RESET).
        25.1.13. Destination (DEST).
    25.2. Provide functionality to display data from a ground API to the on-board DEU CRT (KSC-LPS-OP-03304 section 11.3 (RECORD DATA)).
26. OBS shall provide the functionality to issue commands to the KU-Band Communications Radar or to payload systems that have Payload Signal Processors/Payload Interrogator (PSP/PI) or other special IO device interfaces and to issue commands to the Space-Lab subsystem or experimental computer (KSC-LPS-OP-033-4 section 14.1 (COMMAND PAYLOAD)).
27. OBS shall provide the functionality to command the Payload Data Interleaver (PDI) to perform the PDI Wrap Test (KSC-LPS-OP-033-4 section 14.2 (COMMAND PDI)).
28. OBS shall provide the functionality to read data from the PDI (KSC-LPS-OP-033-4 section 14.3 (READ PDI)).
29. OBS shall provide the functionality to issue CIE data and/or commands to the PCM uplink gateway for 128 KBS forward link command issuance (KSC-LPS-OP-033-4 section 14.4 (COMMAND CIE)).
30. OBS shall provide the functionality to (KSC-LPS-OP-033-4 section ()):
    30.1. Activate/Inhibit Responses.
    30.2. Activate/Inhibit MEC read BITE.
    30.3. Activate/Inhibit read BITE.

4.2.2.2 Gateway Interface Services (GIS) Functional Requirements

1. FD Commanding.
    1.1. GIS shall provide support for an CLCS application to issue quantity type data to a GSE gateway (APPLY ANALOG).

1.1.1. For a single analog command applied to a single analog FD.
1.1.2. For a single analog command applied to many analog FDs.
1.1.3. *For many analog commands applied to many analog FDs.*

1.2. GIS shall provide support for an CLCS application to set discrete statements to a GSE gateway (SET DISCRETE).
1.2.1. For a single discrete command applied to a single discrete FD.
1.2.2. For a single discrete command applied to many discrete FDs.
1.2.3. *For many discrete commands applied to many discrete FDs.*
1.2.4. *To specify a time value to set the command to the indicated state for the specified period and then return it to the original state.*

1.3. GIS shall provide support for an CLCS application to issue a value to digital pattern output FDs to a GSE gateway (ISSUE DIGITAL PATTERNS).
1.3.1. For a single digital pattern command applied to a single digital pattern FD.
1.3.2. For a single digital pattern command applied to many digital pattern FDs.
1.3.3. *For many digital pattern commands applied to many digital pattern FDs.*

1.4. GIS shall provide support for modification of calibration coefficients of PCM and GSE FDs.

1.5. GIS shall provide support for reading calibration coefficients of PCM and GSE FDs.

1.6. GIS shall provide support for an CLCS application to change the sample rate of any GSE FD.

1.7. GIS shall provide a method to activate and inhibit change processing on a per FD basis for a GSE and PCM gateway.

1.8. GIS shall provide a method to activate and inhibit command issuance on a per FD basis by any GSE gateway.

1.9. GIS shall provide a method of reading the current sample rate of an FD.

1.10. GIS shall provide the capability to read the current hardware address of an FD.

1.11. GIS shall provide change hardware address of an FD at any GSE gateway.

1.12. GIS shall provide a method to read the FD directly from the GSE HIM's output.

2. Non-FD Commanding.

2.1. GIS shall provide support for enable and disable PCM and GSE gateway processing.

2.2. GIS shall provide support for changing Sync Bits in Error Count on a PCM gateway.

2.3. GIS shall provide a method to activate and inhibit change processing on a PCM and GSE gateways.

2.4. GIS shall provide a method to activate and inhibit frame logging on a PCM gateway.

2.5. GIS shall provide a method to activate and inhibit data acquisition at any GSE and PCM gateway.

2.6. GIS shall provide a method to activate and inhibit data processing at any GSE and PCM gateway.

2.7. GIS shall provide a method to activate and inhibit command issuance on a GSE gateway.

2.8. GIS shall provide a method to activate and inhibit HIM testing Command.

2.9. GIS shall provide a method to activate and inhibit HIM polling command.

2.10. GIS shall provide support for changing or selecting the active PCM and GSE gateways.

2.11. GIS shall provide a method to read the data acquisition status of any GSE gateway.

4.2.2.3 Non-Gateway Services (NGS) Functional Requirements

1. NGS shall provide a method to publish values to pseudo FDs.
2. NGS shall provide a method to publish health values for FDs using Command Management support.
3. NGS shall provide a method to publish time values for CDT and MET FDs using Command Management support for:
   3.1. Set CDT/MET to a value.
   3.2. Start CDT/MET.
   3.3. Start CDT/MET at a specified GMT/CDT time.
   3.4. Hold CDT/MET.
   3.5. Hold CDT/MET at a specified GMT/CDT time.

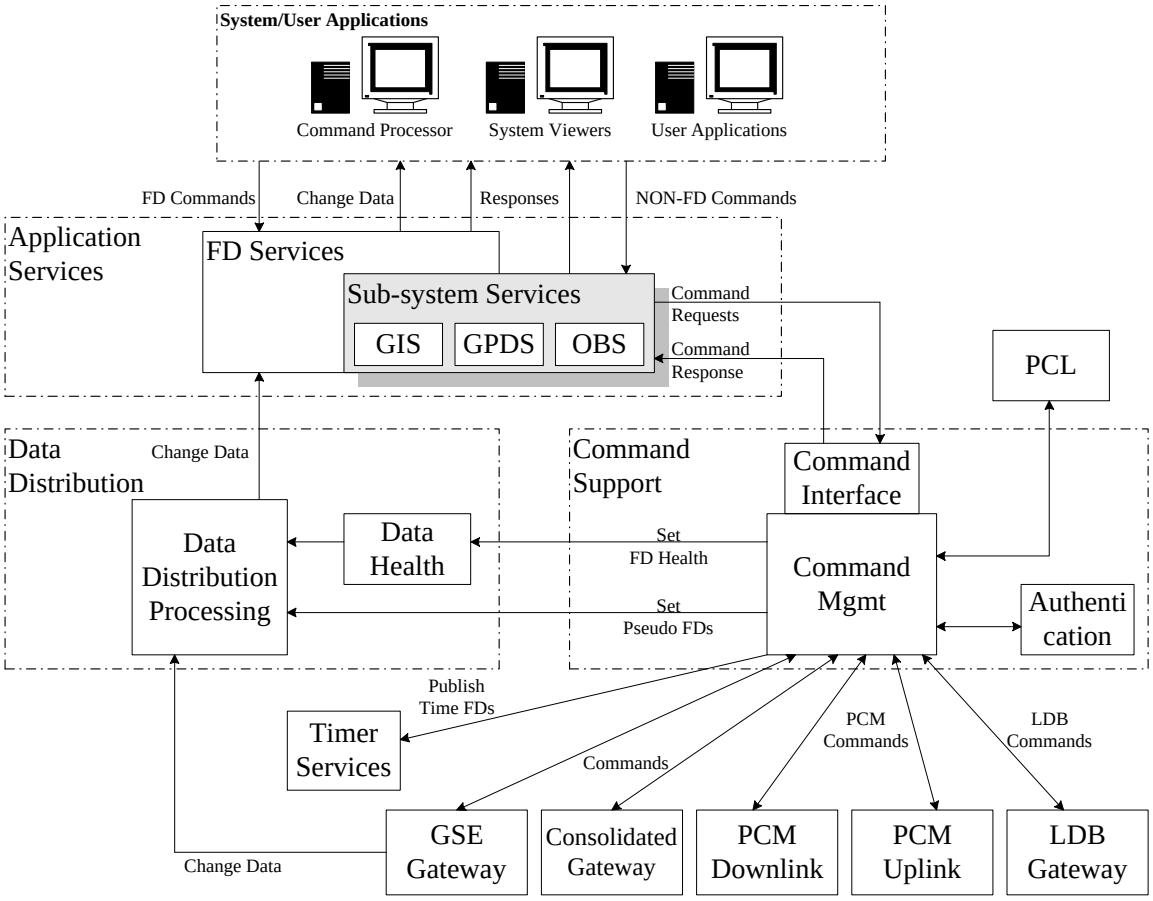
3.6. Cancel pending CDT/MET command.

4.2.3 Sub-system Services Performance Requirements

There are no performance requirements for Sub-system Services.

4.2.4 Sub-system Services CSC Interfaces Data Flow Diagrams

External Data Flow Diagram Example



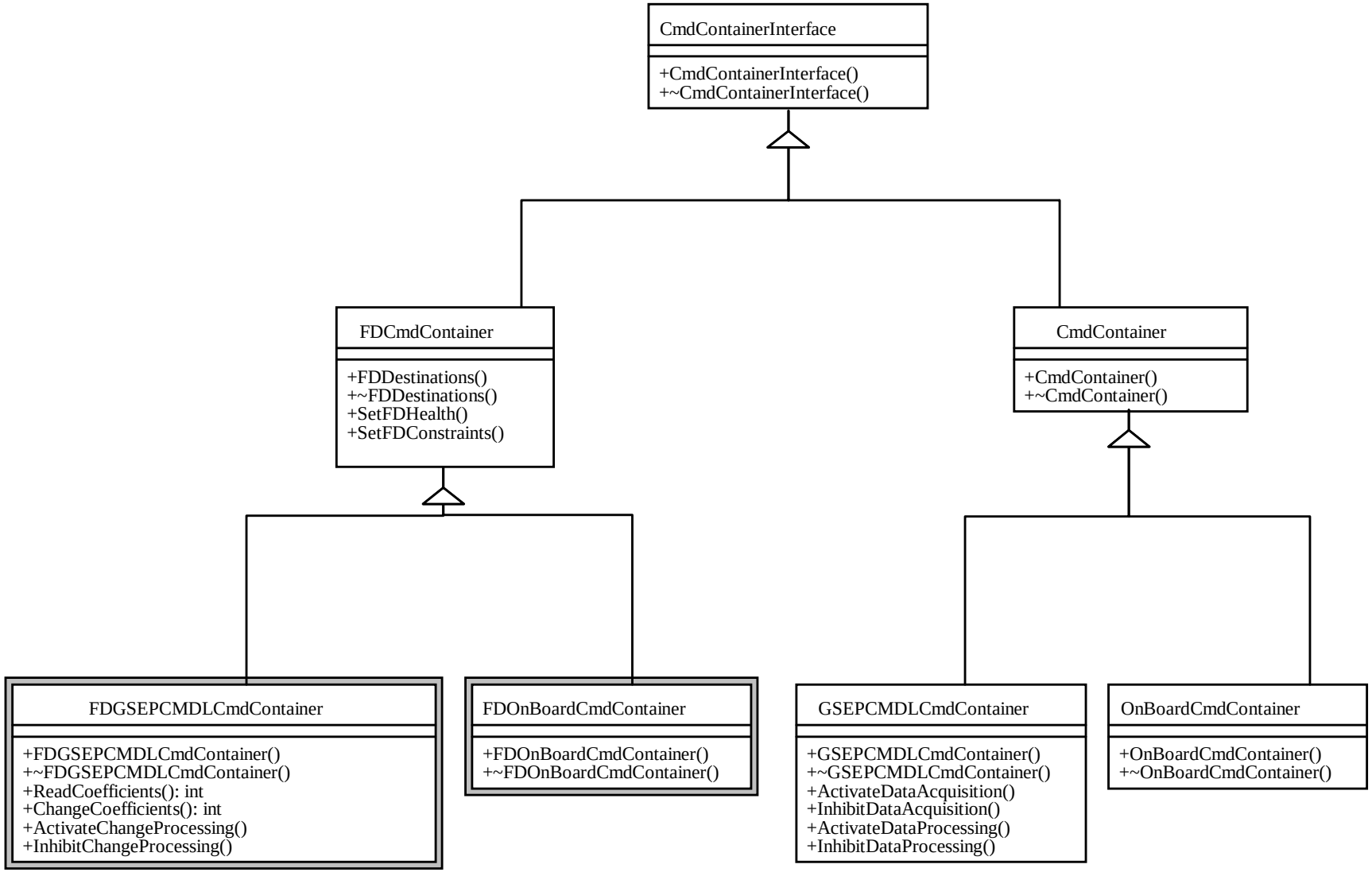
Sub-system services consist of objects tied into FD Services for a seamless integration of FD commands. When an application wants to perform an FD command, they will call an API through the FD object. The FD object will then have SSS package the command and send it out to Command Interface portion of Command Management. Command Interface will build the necessary packet to be sent out to the logical destination. Command Management will call the necessary Prerequisite Control Logic (PCL) and authenticate the command. The response code will be sent back to SSS to provide it for the FD object. If the command is associated with an FD object not assigned to a gateway, then Command Management will send it to the appropriate service to process the FD information as a command. If the command is not FD related, then the user will be able to create an object for the particular sub-system to call a command that is only associated with the appropriate destination. SSS will then package the command and send it to Command Interface to put the information into a packet to be sent through Command Management to the logical destination. Responses will then be sent back to SSS, and the service will provide any necessary conversions of the data to provide for the users similar capabilities found in the Ground Operations Aerospace Language (GOAL).

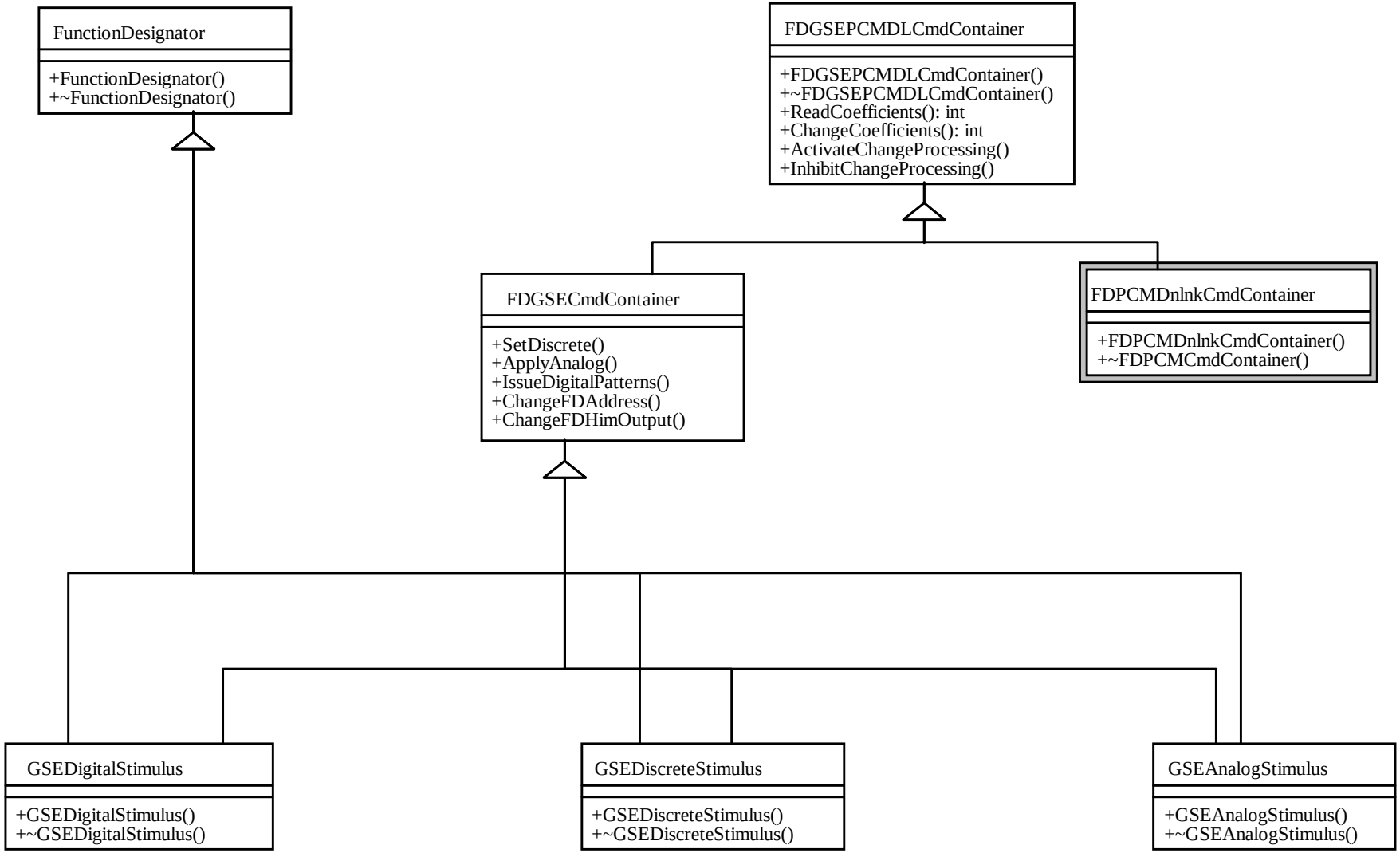
4.3 Sub-system Services Design Specification

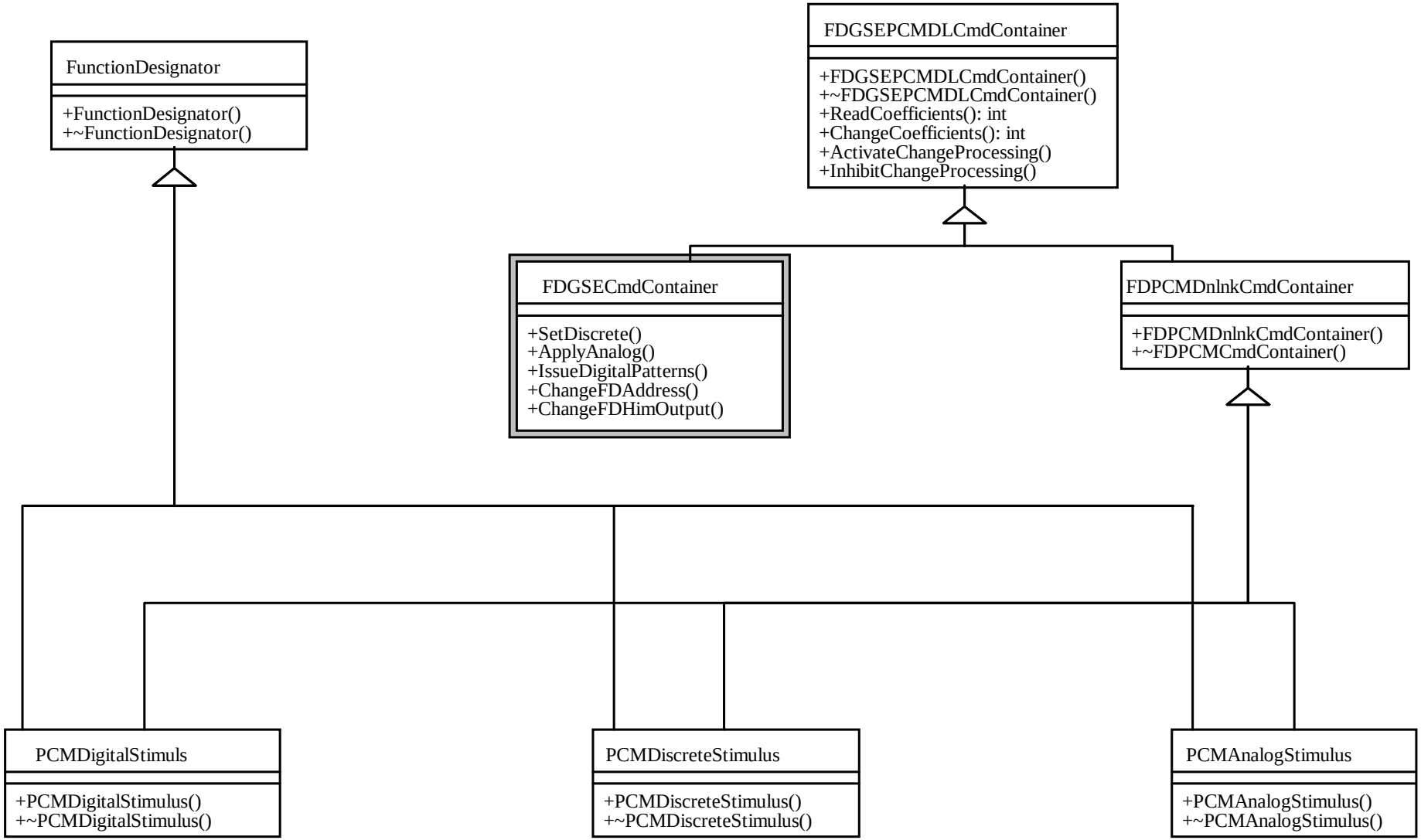
4.3.1 Sub-system Services Detailed Data Flow

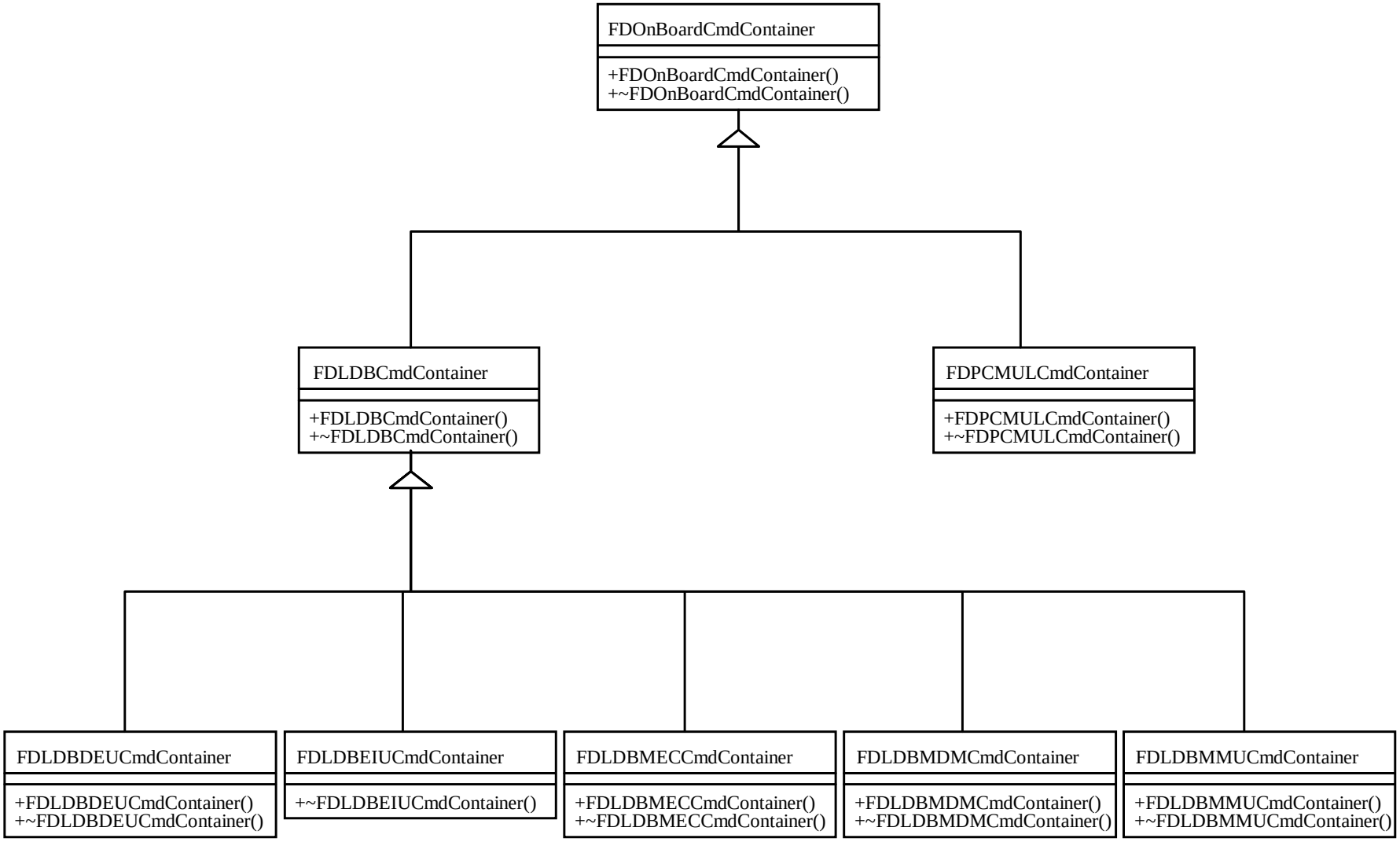
TBS.

Detailed Data Flow Diagram


CmdContainerInterface
+CmdContainerInterface()
+~CmdContainerInterface()
FDCmdContainer
+FDDestinations()
+~FDDestinations()
+SetFDHealth()
+SetFDConstraints()
CmdContainer
+CmdContainer()
+~CmdContainer()
FDGSEPCMDLCmdContainer
+FDGSEPCMDLCmdContainer()
+~FDGSEPCMDLCmdContainer()
+ReadCoefficients(): int
+ChangeCoefficients(): int
+ActivateChangeProcessing()
+InhibitChangeProcessing()
FDOnBoardCmdContainer
+FDOnBoardCmdContainer()
+~FDOnBoardCmdContainer()
GSEPCMDLCmdContainer
+GSEPCMDLCmdContainer()
+~GSEPCMDLCmdContainer()
+ActivateDataAcquisition()
+InhibitDataAcquisition()
+ActivateDataProcessing()
+InhibitDataProcessing()
OnBoardCmdContainer
+OnBoardCmdContainer()
+~OnBoardCmdContainer()


FunctionDesignator
+FunctionDesignator()
+~FunctionDesignator()
FDGSEPCMDLCmdContainer
+FDGSEPCMDLCmdContainer()
+~FDGSEPCMDLCmdContainer()
+ReadCoefficients(): int
+ChangeCoefficients(): int
+ActivateChangeProcessing()
+InhibitChangeProcessing()
FDGSECmdContainer
+SetDiscrete()
+ApplyAnalog()
+IssueDigitalPatterns()
+ChangeFDAddress()
+ChangeFDHimOutput()
FDPCMDnlnkCmdContainer
+FDPCMDnlnkCmdContainer()
+~FDPCMCmdContainer()
GSEDigitalStimulus
+GSEDigitalStimulus()
+~GSEDigitalStimulus()
GSEDiscreteStimulus
+GSEDiscreteStimulus()
+~GSEDiscreteStimulus()
GSEAnalogStimulus
+GSEAnalogStimulus()
+~GSEAnalogStimulus()


FunctionDesignator
+FunctionDesignator()
+~FunctionDesignator()
FDGSEPCMDLCmdContainer
+FDGSEPCMDLCmdContainer()
+~FDGSEPCMDLCmdContainer()
+ReadCoefficients(): int
+ChangeCoefficients(): int
+ActivateChangeProcessing()
+InhibitChangeProcessing()
FDGSECmdContainer
+SetDiscrete()
+ApplyAnalog()
+IssueDigitalPatterns()
+ChangeFDAddress()
+ChangeFDHimOutput()
FDPCMDnlnkCmdContainer
+FDPCMDnlnkCmdContainer()
+~FDPCMCmdContainer()
PCMDigitalStimuls
+PCMDigitalStimulus()
+~PCMDigitalStimulus()
PCMDiscreteStimulus
+PCMDiscreteStimulus()
+~PCMDiscreteStimulus()
PCMAnalogStimulus
+PCMAnalogStimulus()
+~PCMAnalogStimulus()



SSS Class Diagrams

4.3.2 Sub-system Services External Interfaces

4.3.2.1 Sub-system Services Message Formats

TBD.

4.3.2.2 Sub-system Services Display Formats

Not applicable. SSS does not provide any displays.

4.3.2.3 Sub-system Services Input Formats

No applicable. There are no language-like interfaces provided by SSS.

4.3.2.4 Recorded Data

SSS does not record data nor initiate data recording.

4.3.2.5 Sub-system Services Printer Formats

SSS does not provide printed information.

4.3.2.6 Interprocess Communications (C-to-C Communications?)

Not applicable.

4.3.2.7 Sub-system Services External Interface Calls

The CLCS SSS Interface Description Document describes the data sent between the SSS CSC and CLCS applications via a calling mechanism. The CLCS FD Command Interface Description Document 84K00352 describes the interface between SSS and Command Management.

4.3.2.8 Sub-system Services Name Table Formats

SSS does not utilize any tables internally that are provided from an outside source.

4.3.3 Sub-systems Services Test Plan

The test plan provided by Jack Blackledge, John Wilkenson, and Tom Jamieson for Commanding End to End will be used for SSS.